FACULTADE DE MATEMÁTICAS

Traballo Fin de Grao

Optimización con restricciones y control óptimo de sistemas discretos

Sandra Díaz Seoane

2019/2020

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

GRAO DE MATEMÁTICAS

Traballo Fin de Grao

Optimización con restricciones y control óptimo de sistemas discretos

Sandra Díaz Seoane

Septiembre,2020

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Trabajo propuesto

Área de Coñecemento: Matemática aplicada
Título: Optimización con restricciones y control óptimo de sistemas discretos
Breve descrición do contido
Nos interesamos por problemas de optimización del tipo Minimizar $\tilde{J}(u)$ $u \in U$ donde u, la variable de minimización o variable de control, pertenece a un conjunto de controles llamados admisibles. Con frecuencia la expresión del funcional $\tilde{J}(u)$ no admite una expresión analítica sencilla pero sin embargo verifica $\tilde{J}(u) = J(y(u), u)$, donde $y(u)$, llamada la variable de estado, es la única solución de una ecuación que representamos por $F(y, u) = 0$, y que recibe el nombre de sistema de estado. Los problemas de optimización con estas características reciben el nombre de problemas de control óptimo y debido a su estructura particular, admiten técnicas de análisis y resolución especiales. Nos centraremos en el caso en el que tanto la variable de estado como la de control son vectores y el sistema de estado vendrá dado por un sistema de ecuaciones algebraicas. En el trabajo se particularizarán los conceptos propios de problemas de optimización (tales como condiciones de optimalidad o el cálculo del gradiente del funcional coste) a problemas con esta estructura. Por otra parte, se propondrán e implementarán métodos numéricos sencillos para su resolución numérica.

Índice general

Resumen

En este trabajo se hará un estudio del control óptimo de sistemas discretos. Empezaremos haciendo un repaso de optimización que nos será de gran utilidad para tratar posteriormente el problema de control óptimo de sistemas discretos, por poder entenderse este como un problema de optimización con y sin restricciones. Se tratará el estudio de las condiciones para la existencia de mínimos así como se darán diferentes métodos de resolución para los distintos tipos de problemas. Se aplicará lo visto en optimización al control óptimo, adaptándolo a las características especiales de estos problemas. Se aplicarán dichas técnicas al aprendizaje de una red neuronal de clasificación. Finalmente se dará un caso particular de control óptimo de sistemas discretos procedente de una discretización de un problema de control óptimo con EDO.

Abstract

This thesis is devoted to the study of optimal control in the framework of discrete systems. As a first step we will review some basic results on mathematical optimization, namely necessary and sufficient conditions for existence and uniqueness of extrema as well as several numerical methods for its computation. These results and methods will be adapted and applied to optimal control problems arising from discrete systems. Some of these techniques will be implemented in Matlab. The codes will be validated solving some academic control problems. Finally, these techniques will be also used to train an artificial neural network and to solve control problems for discrete dynamical systems issued from the discretion of optimal control problems for ODE.

Introducción

En este trabajo hablaremos de control óptimo de sistemas discretos que consistirá en la optimización de un sistema que evoluciona a lo largo de unos instantes y que se verá influenciado por el valor de ciertas variables. Se aplicará, por ejemplo, en la evolución del cuerpo humano, intentando alcanzar un cierto rendimiento o forma física; en la economía, buscado alcanzar por ejemplo un deficit concreto o unos beneficios; o en la física, relacionado por ejemplo con la conducción del calor al intentar mantener una habitación a una cierta temperatura. También habrá aplicaciones aeroespaciales, robóticas, aeronáuticas, en las redes de comunicación o en la química entre otras.

Para este trabajo será necesario el uso de conocimientos de las áreas de álgebra lineal y del análisis matemático, principalmente enfocados en el estudio de las existencia de mínimos de los distintos tipos de problemas que se ven a lo largo del texto, en este ámbito usaremos como referencias los apuntes de las materias cursadas en la carrera así como [1], [2] y [3]. Sin embargo el peso más importante está en los métodos numéricos en optimización.

Los dos primeros capítulos del trabajo se basa en conceptos de optimización sin restricciones y con restricciones que se han estudiado parcialmente en la asignatura de "Métodos numéricos en optimización y ecuaciones diferenciales". En la preparación de estos contenidos se han consultado fundamentalmente las referencias [4] y [5] así como las notas del curso de la materia.

Para la preparación del tercer capítulo que se centra en el control óptimo de sistemas discretos tendremos como referencias [6], [7] y [8]. Será quizás la última la más usada y la primera solo consultada como apoyo para entender mejor los conceptos.

Este documento se estructura en tres capítulos bien diferenciados. En el primero de ellos se proporciona un repaso de optimización sin restricciones, llevando a cabo inicialmente una introducción de conceptos y resultados de álgebra lineal y análisis matemático que como ya dijimos antes serán de interés especialmente en el estudio de los extremos de una función. A continuación nos centramos en los métodos numéricos en optimización sin restricciones. Comenzaremos inicialmente con la descripción y análisis de un algoritmo genérico de descenso y las reglas de cálculo de paso. A continuación se verán distintas

variantes del mismo tales como el método del gradiente, el de Newton, gradiente conjugado y cuasi-Newton.

El segundo capítulo se dedica a la optimización con restricciones. En este caso empezaremos con un estudio del caso en que solo hay restricciones de igualdad. Daremos una descripción del problema, estudiaremos brevemente condiciones de optimalidad basada en los multiplicadores de Lagrange y la resolución con el método de penalización. Seguiremos entonces con el estudio de la optimización con restricciones de igualdad y desigualdad donde haremos un estudio más pormenorizado. Nuevamente nos centraremos en las condiciones para la existencia de mínimos tanto de primer orden como de segundo orden y daremos un estudio un poco más riguroso del método de penalización.

Finalmente, el tercer capítulo del trabajo se centra en el control óptimo de sistemas discretos. Comenzaremos definiendo el problema para a continuación hablar sobre su planteamiento como problema de optimización con y sin restricciones. En el caso del planteamiento con restricciones, estás serán de igualdad y veremos principalmente como se aplican a este problema las condiciones basadas en los multiplicadores de Lagrange vistas en el capítulo 2. En el planteamiento sin restricciones nos centraremos en el cálculo del gradiente. Motivaremos por qué es interesante obtenerlo y daremos dos métodos para hacerlo. Hecho esto veremos la resolución tratándolo como un problema de optimización sin restricciones. Daremos un código en el que se incluirán las dos opciones para el cálculo del gradiente así como algunos métodos de los vistos en el capítulo 1 y los probaremos con algunos ejemplos académicos. Para concluir con el capítulos veremos aplicaciones de este tipo de problemas. Comenzaremos hablando de sistemas dinámicos discretos y como al aplicarles el control óptimo obtenemos un método propio de resolución. Seguiremos con la aplicación de lo anterior al aprendizaje de una red neuronal de clasificación basada en el artículo [9], viendo su formulación matemática, como se le aplica el método de resolución y un código generalizando el dado en el artículo. Para terminar veremos como se aplicaría lo estudiado a un problema basado en la discretización de un problema de control óptimo con EDO.

Capítulo 1

Optimización sin restricciones

En esta sección haremos una introducción a los problemas de optimización, centrándonos en aquellos sin restricciones. Veremos definiciones y relaciones interesantes entre ellas. Nos servirán para delimitar bajo qué condiciones estos problema tienen solución y cuándo va a ser única. También nos centraremos en los métodos de resolución y su implementación. Posteriormente veremos que en el capítulo 2 el método que daremos para solucionar los problemas consistirá a grandes rasgos en la resolución de un problema de optimización sin restricciones para lo cual usaremos lo visto a continuación.

1.1. Preliminares

Vamos a hacer un repaso de definiciones y resultados de álgebra lineal y análisis que nos van a ser útiles en el estudio de la optimización. En primer lugar veremos aquellos del álgebra lineal básicamente relacionados con matrices empezando por autovalores y autovectores.

Definición 1.1. Dada $A \in \mathbb{R}^{n\times n}$ matriz, diremos $\lambda \in \mathbb{R}$ es un *autovalor* si existe $x \in \mathbb{R}^n \setminus \{0\}$ tal que $Ax = \lambda x$. En este caso, se dice que x es un *autovector* asociado al autovalor λ.

Proposición 1.2. *Sea $A \in \mathbb{R}^{n\times n}$ con autovalores $\lambda_1, \ldots, \lambda_n$.*

1. *Se pueden seleccionar autovectores asociados a los distintos autovalores de A que son linealmente independientes entre si.*
2. *A es diagonalizable si, y solo si, para cada autovalor de A, su multiplicidad geométrica es igual a la algebraica, i.e., la dimensión del espacio vectorial engendrado por los correspondientes autovectores es igual a la multiplicidad de los autovalores como raices del polinomio característico $p(\lambda) = det(A - \lambda I)$.*

Demostración.

1. Véase la proposición 6.2.5, capítulo 6, página 216 del libro [1].

2. Véase el resultado 173, capítulo 9, página 347 del libro [2].

□

Vemos ahora resultados relacionados con matrices simétricas ya que habrá menciones a ellas a lo largo de este texto. Vemos posteriormente la definición, así como relaciones con autovalores y autovectores, de matriz definida positiva y demás. Estos conceptos se usarán en resultados en los cuales se habla de condiciones necesarias y suficientes para la existencia de mínimos (Teoremas 1.23 y 1.24).

Proposición 1.3. *(Propiedades de las matrices simétricas)*
Sea $A \in \mathbb{R}^{n\times n}$ simétrica. Entonces

1. *Todos los autovalores de A son reales.*

2. *A es ortogonalmente similar a una matriz diagonal, i.e., existe una matriz ortogonal $Q \in \mathbb{R}^{n\times n}$ tal que*

$$Q^{-1}AQ = Q^TAQ = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix},$$

con $\lambda_1, \ldots, \lambda_n$ autovalores de A.

Demostración.

Véase el capítulo 8, teorema 8.4.7, página 380 del libro [1].

Véase el Capítulo 8, teorema 8.4.8, página 380 del libro [1]. □

Definición 1.4. Sea $A \in \mathbb{R}^{n\times n}$ simétrica. A se dice *definida positiva* si $v^TAv > 0, \forall v \in \mathbb{R}^n$, $v \neq 0$. Se dice que es *semidefinida positiva* si $v^TAv \geq 0, \forall v \in \mathbb{R}^n$. A es *definida negativa* o *semidefinida negativa* si $-A$ es definida positiva o semidefinida positiva respectivamente. A se dice *indefinida* si no es ni semidefinida positiva ni semidefinida negativa.

Dada la definición 1.4 tenemos a partir de la proposición 1.3 el siguiente corolario.

Corolario 1.5. *(Matrices definidas y semidefinidas positivas y negativas e indefinidas)*
Sea $A \in \mathbb{R}^{n\times n}$ simétrica:

1. *A es definida positiva $\Leftrightarrow$ todos sus autovalores son positivos.*

2. *A es semidefinida positiva $\Leftrightarrow$ todos sus autovalores son no negativos.*

3. *A es definida negativa o semidefinida negativa si todos sus autovalores son negativos o no positivos respectivamente.*

4. *A es entonces indefinida si tiene autovalores tanto positivos como negativos.*

Veremos ahora las definiciones y resultados relacionados con el análisis matemático. Comenzamos con definiciones de función continua, continuamente diferencianble y dos veces continuamente diferenciable.

Definición 1.6. Una función $f : \mathbb{R}^n \to \mathbb{R}$ se dice *continua en* $\hat{x} \in \mathbb{R}^n$ si, para cada $\epsilon > 0$, existe un $\delta > 0$ tal que $\|x - \hat{x}\| < \delta$ implica $\|f(x) - f(\hat{x})\| < \epsilon$. Si f es continua en cada punto de un conjunto abierto $B \subset \mathbb{R}^n$, entonces de dice que f es *continua en* B, $f \in \mathscr{C}(B)$.

Definición 1.7. Una función continua f se dice *continuamente diferenciable en* $x \in \mathbb{R}^n$, si $\frac{\partial f}{\partial x_i}(x)$ existe y es continua para $i = 1, \ldots, n$. El *gradiente* de f en x se define como

$$\nabla f(x) = \left[\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x)\right]^T.$$

Si f es continuamente diferenciable en todo punto de un conjunto $B \subset \mathbb{R}^n$, entonces se dice que f es *continuamente diferenciable en* B, $f \in \mathscr{C}^1(B)$.

Definición 1.8. Una función continuamente diferenciable f se dice *dos veces continuamente diferenciable en* $x \in \mathbb{R}^n$, si $\frac{\partial^2 f}{\partial x_i \partial x_j}(x)$ existen y son continuas para $1 \leq i, j \leq n$. La *Hessiana* de f en x se define como la matriz $n \times n$ cuyos elementos son

$$\left[\nabla^2 f(x)\right]_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}(x), 1 \leq i, j \leq n.$$

Denotaremos también a la Hessiana de f en x como $Hf(x)$. Si f es dos veces continuamente diferenciable en todo punto de un conjunto $B \subset \mathbb{R}^n$, entonces se dice que f es *dos veces continuamente diferenciable en* B, $f \in \mathscr{C}^2(B)$.

Observación 1.9. El teorema de Schwarz que podemos encontrar en el capítulo 5, teorema 5.6, página 194 de [3] afirma que la matriz hessiana de una función dos veces continuamente diferenciable es simétrica.

Finalizando esta sección tenemos definiciones de convexidad tanto para conjuntos como para funciones, así como resultados relacionados con estás. En primer lugar la definición y resultados de conjuntos convexos.

Definición 1.10. Sea $S \subset \mathbb{R}^n$. S es *convexo* si $\forall x, y \in S, \quad \lambda x + (1-\lambda)y \in S, \quad \forall \lambda \in [0, 1]$.

Observación 1.11. Geométricamente la definición anterior nos indica que para cada dos puntos del conjunto el segmento que los une está contenido en dicho conjunto. Por lo cual vemos que todo conjunto convexo es conexo.

Proposición 1.12. *Sean $S, S_1, S_2 \in \mathbb{R}^n$ conjuntos convexos. Entonces:*

1. *$S_1 \bigcap S_2$ es convexo.*
2. *$S_1 \pm S_2 = \{x_1 \pm x_2 | x_1 \in S_1, x_2 \in S_2\}$ es convexo.*
3. *El interior de S es convexo.*
4. *La clausura de S, $\bar{S}$, es un conjunto convexo.*

Demostración. 1) Sean $x, y \in S_1 \bigcap S_2$. Tenemos que $x, y \in S_1$ y por ser S_1 convexo tenemos que $\lambda x + (1-\lambda)y \in S_1, \quad \forall \lambda \in [0,1]$. Análogamente para S_2 tenemos $\lambda x + (1-\lambda)y \in S_2, \ \forall \lambda \in [0,1]$. Entonces $\lambda x + (1-\lambda)y \in S_1 \bigcap S_2, \ \forall \lambda \in [0,1]$ y hemos demostrado que $S_1 \bigcap S_2$ es convexo.

2) Sean $x, y \in S_1 \pm S_2$. Tenemos $x = x_1 \pm x_2$ para ciertos $x_1 \in S_1$, $x_2 \in S_2$ e $y = y_1 \pm y_2$ para ciertos $y_1 \in S_1$, $y_2 \in S_2$. Por ser S_1 convexo tenemos $\lambda x_1 + (1-\lambda)y_1 \in S_1, \forall \lambda \in [0,1]$; análogamente para S_2; $\lambda x_2 + (1-\lambda)y_2 \in S_2, \forall \lambda \in [0,1]$. Entonces tenemos

$$\lambda x_1 + (1-\lambda)y_1 \pm \lambda x_2 + (1-\lambda)y_2 = \lambda(x_1 \pm x_2) + (1-\lambda)(y_1 \pm y_2) = \lambda x + (1-\lambda)y \in S_1 \pm S_2.$$

Con lo cual queda demostrado que $S_1 \pm S_2$ es convexo.

3) Sean x e y dos puntos en el interior de S y $z = \lambda x + (1-\lambda)y$ con $\lambda \in [0,1]$. Escogemos un $\delta > 0$ tal que $B(y, \delta) \subset S$, siendo $B(y, \delta)$ el entorno de y con radio δ. Tenemos $\frac{||z-x||}{||y-x||} = 1 - \lambda$. Entonces $B(z, (1-\lambda)\delta)$ es el conjunto $\lambda x + (1-\lambda)B(y, \delta)$ que está contenido en S. Finalmente $B(z, (1-\lambda)\delta) \subset S$ prueba que z pertenece al interior de S y que este es convexo.

4) Sean $x, y \in \bar{S}$ escogemos en S dos sucesiones x_k e y_k convergente a x e y respectivamente. Entonces sea $\lambda \in [0,1]$ tenemos

$$\begin{aligned} ||[\lambda x_k + (1-\lambda)y_k] - [\lambda x + (1-\lambda)y]|| &= ||\lambda(x_k - x) + (1-\lambda)(y_k - y)|| \\ &\leq \lambda||x_k - x|| + (1-\lambda)||y_k - y||. \end{aligned}$$

Haciendo el límite tenemos

$$\lim_{k \to \infty} ||[\lambda x_k + (1-\lambda)y_k] - [\lambda x + (1-\lambda)y]|| = 0,$$

lo cual demuestra que $\lambda x + (1-\lambda)y \in \bar{S}$, es decir, $\bar{S}$ es convexo.

□

Seguimos ahora con funciones convexas.

Definición 1.13. Sea $S \subset \mathbb{R}^n$ un conjunto no vacío y convexo. La función $f : S \to \mathbb{R}$ es *convexa* si:

$$\forall x, y \in S, \quad f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y), \quad \forall \lambda \in [0,1],$$

y es *estrictamente convexa* si:

$$\forall x, y \in S \text{ tales que } x \neq y, \quad f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda) f(y), \quad \forall \lambda \in (0,1).$$

Proposición 1.14.
Dadas $J_1, \ldots, J_n$ funciones convexas y los números reales $\alpha_1 \geq 0, \ldots, \alpha_n \geq 0$, entonces $\sum_{i=1}^{n} \alpha_i J_i$ es una función convexa.

Demostración. Sean $x, y \in S$ y $\lambda \in [0,1]$, entonces

$$\sum_{i=1}^{n} \alpha_i J_i(\lambda x + (1-\lambda)y) \leq \sum_{i=1}^{n} \alpha_i \lambda J_i(x) + \alpha_i (1-\lambda) J_i(y)$$

$$= \lambda \sum_{i=1}^{n} \alpha_i J_i(x) + (1-\lambda) \sum_{i=1}^{n} \alpha_i J_i(y).$$

□

Se tienen los resultados siguientes cuyas demostraciones las encontraremos en el libro [4].

Proposición 1.15. *Sean $S \subset \mathbb{R}^n$ abierto y convexo y $f : S \subset \mathbb{R}^n \to \mathbb{R}$ convexa. Entonces f es continua en S.*

Demostración. Véase el capítulo 1, teorema 1.3.12, página 40 de [4]. □

Proposición 1.16. *Sea S abierto y convexo, $f : S \to \mathbb{R}$ diferenciable. Entonces f es convexa si, y solo si, $f(y) \geq f(x) + \nabla f(x)^t (y - x)$, $\forall x, y \in S$. Además f es estrictamente convexa si, y solo si, $f(y) > f(x) + \nabla f(x)^t (y - x)$, $\forall x, y \in S$ con $y \neq x$.*

Demostración. Véase el capítulo 1, teorema 1.3.13, página 42 de [4]. □

Proposición 1.17. *Sea S abierto y convexo, $f : S \to \mathbb{R}$, $f \in \mathscr{C}^2(S)$. Entonces f es convexa si, y solo si, $Hf(x)$ es semidefinida positiva $\forall x \in S$. Además f es estrictamente convexa si $Hf(x)$ es definida positiva $\forall x \in S$.*

Demostración. Véase el capítulo 1, teorema 1.3.14, página 44 de [4]. □

Observación 1.18. Diremos que $f : \mathbb{R}^n \to \mathbb{R}$ es *cuadrática* si $f(x) = \frac{1}{2}x^T Ax + x^T b + c$ con $A \in \mathbb{R}^{n\times n}$ simétrica, $b \in \mathbb{R}^n$ y c un número real. Para este tipo de funciones, tenemos:

1. $\nabla f(x) = Ax + b, \quad Hf(x) = A \quad \forall x \in \mathbb{R}^n$.
2. f en convexa $\Leftrightarrow$ A es semidefinida positiva.
3. f es estrictamente convexa $\Leftrightarrow$ A es definida positiva.

1.2. Condiciones necesarias y suficientes para la existencia de mínimos

La forma general de un problema de optimización sin restricciones es

$$\min_{x\in X} J(x)$$

donde $x \in \mathbb{R}^n$ es la variable de optimización, $J(x)$ la función objetivo y $X \equiv \mathbb{R}^n$ su conjunto o región factible.

Vamos a realizar un estudio de las condiciones necesarias y suficientes para la existencia de mínimos locales y globales. Comenzamos definiendo estos mínimos.

Definición 1.19. Un punto $\hat{x}$ se llama *mínimo local* si $\exists\delta > 0$ tal que $J(\hat{x}) \leq J(x)\ \forall x \in \mathbb{R}^n$ que satisface $||x - \hat{x}|| < \delta$. Un punto $\hat{x}$ se llama *mínimo local estricto* si $\exists\delta > 0$ tal que $J(\hat{x}) < J(x)\ \forall x \in \mathbb{R}^n$ con $x \neq \hat{x}$ y $||x - \hat{x}|| < \delta$.

Definición 1.20. Sea $\hat{x} \in X$, un mínimo local tal que existe un entorno de $\hat{x}$, $B(\hat{x}, \delta)$ con $\delta > 0$, en el cual $\hat{x}$ es el único mínimo local en $X \bigcap B(\hat{x}, \delta)$; entonces diremos que $\hat{x}$ es un *mínimo local aislado*.

Definición 1.21. Un punto $\hat{x}$ se llama *mínimo global* si $J(\hat{x}) \leq J(x)\ \forall x \in \mathbb{R}^n$. Un punto $\hat{x}$ se llama *mínimo global estricto* si $J(\hat{x}) < J(x)\ \forall x \in \mathbb{R}^n$ con $x \neq \hat{x}$.

Teorema 1.22 (Condición necesaria de primer orden)**.** *Sea $J : D \subset \mathbb{R}^n \to \mathbb{R}$ continuamente diferenciable en el abierto D, $J \in \mathscr{C}^1(D)$. Si $\hat{x} \in D$ es mínimo local de J, entonces $\nabla J(\hat{x}) = 0$.*

Demostración. Sea $\hat{x} \in D$ un mínimo local y consideramos la sucesión $x_k = \hat{x} - \alpha_k \bigtriangledown J(\hat{x})$, $\alpha_k > 0$, con $\{\alpha_k\}$ tendiendo a cero. Usando el desarrollo de Taylor, para k suficientemente grande,

$$0 \leq J(x_k) - J(\hat{x}) = -\alpha_k \nabla J(\eta_k)^T \nabla J(\hat{x}),$$

donde η_k es una combinación convexa de x_k y $\hat{x}$. Ahora bien pasando al límite cuando x_k tiende a $\hat{x}$ después de haber dividido por α_k, se sigue de $J \in \mathscr{C}^1(D)$ que $0 \leq -||\nabla J(\hat{x})||^2$ lo cual quiere decir que $\nabla J(\hat{x}) = 0$. □

Seguimos ahora con condiciones necesarias de primer y segundo orden para funciones generales.

Teorema 1.23 (Condición necesaria de segundo orden)**.** *Sea $J : D \subset \mathbb{R}^n \to \mathbb{R}$, D un abierto y $J \in \mathscr{C}^2(D)$. Si $\hat{x} \in D$ es mínimo local de J, entonces $\nabla J(\hat{x}) = 0$ y $Hf(\hat{x})$ es semidefinida positiva.*

Demostración. Por el teorema anterior tenemos que $\nabla J(\hat{x}) = 0$ por tanto, solo tenemos que demostrar que $HJ(\hat{x})$ es semidefinida positiva. Consideramos la sucesión $x_k = \hat{x} - \alpha_k d$, $\alpha_k > 0$, con $\{\alpha_k\}$ tendiendo a cero, donde d es arbitrario. Como $J \in \mathscr{C}^2(D)$ y $\nabla J(\hat{x}) = 0$, entonces el desarrollo de Taylor que tenemos para k suficientemente grande será

$$0 \leq J(x_k) - J(\hat{x}) = \frac{1}{2}\alpha_k^2 d^T HJ(\eta_k)d,$$

donde η_k es una combinación convexa de x_k y $\hat{x}$. Dividiendo ahora entre $\frac{1}{2}\alpha_k^2$ y pasando al límite, tenemos $d^T HJ(\hat{x})d \geq 0, \quad \forall d \in \mathbb{R}^n$. □

Tenemos a continuación una condición suficiente de segundo orden para funciones generales.

Teorema 1.24 (Condición suficiente de segundo orden)**.** *Sea $J : D \subset \mathbb{R}^n \to \mathbb{R}$, D un abierto y $J \in \mathscr{C}^2(D)$. Si $\nabla J(\hat{x}) = 0$ y $HJ(\hat{x})$ es definida positiva, entonces $\hat{x} \in D$ es mínimo local estricto de J.*

Demostración. Supongamos que $\nabla J(\hat{x}) = 0$ y $HJ(\hat{x})$ es definida positiva. Usando el desarrollo de Taylor, para cualquier vector $d \in \mathbb{R}^n$ tal que $\hat{x} + d$ esté en un entorno de $\hat{x}$ en el cual $HJ(\hat{x} + d)$ es definida positiva, tenemos $J(\hat{x} + d) = J(\hat{x}) + \frac{1}{2}d^T HJ(\hat{x} + \theta d)d$, donde $\theta \in (0, 1)$. Entonces podemos escoger un $\delta > 0$ tal que $\hat{x}+d \in B(\hat{x}, \delta)$ y $d^T HJ(\hat{x}+\theta d)d > 0$. Entonces, $J(\hat{x} + d) > J(\hat{x})$ y tenemos demostrado el teorema. □

Terminamos esta sección con condiciones necesarias y suficientes pero en este caso en relación a funciones convexas. La primera es una condición suficiente para la existencia de un mínimo global y la segunda es necesaria y suficiente.

Proposición 1.25. *Sea $S \subset \mathbb{R}^n$ un conjunto convexo y $J : S \to \mathbb{R}$ una función convexa. Supongamos que existe $\hat{x} \in S$ mínimo local de J. Entonces:*

1. *$\hat{x}$ es mínimo global.*

2. Si J es además estrictamente convexa, $\hat{x}$ es el único mínimo global.

Demostración. 1. Supongamos que existe $x \in S$ tal que $J(x) < J(\hat{x})$. Para todo $\lambda \in (0,1)$ se tiene:

$$J(\lambda x + (1-\lambda)\hat{x}) \leq \lambda J(x) - (1-\lambda)J(\hat{x}) < \lambda J(\hat{x}) + (1-\lambda)J(\hat{x}) = J(\hat{x}).$$

Luego existen puntos tan próximos a $\hat{x}$ como queramos en los que J toma un valor menor que $J(\hat{x})$, lo que contradice el hecho de que $\hat{x}$ sea mínimo local.

2. Supongamos existe $x \in S \backslash \{\hat{x}\}: \quad J(x) = J(\hat{x})$. Para todo $\lambda \in (0,1)$ se tiene:

$$J(\lambda x + (1-\lambda)\hat{x}) < \lambda J(\hat{x}) + (1-\lambda)J(\hat{x}) = J(\hat{x}).$$

Llegamos de nuevo a una contradicción.

□

Teorema 1.26. *Sea $S \subset \mathbb{R}^n$ un conjunto abierto y convexo y $J : S \to \mathbb{R}$ una función diferenciable y convexa. Entonces:*

$$\hat{x} \text{ es mínimo global } \Leftrightarrow \nabla J(\hat{x}) = 0.$$

Demostración. "⇒"

$\hat{x}$ mínimo global $\Rightarrow$ $\hat{x}$ mínimo local $\Rightarrow \nabla J(\hat{x}) = 0$.

"⇐"

Por la Proposición 1.16 sabemos que $J(x) \geq J(\hat{x}) + \nabla J(\hat{x})^T(x - \hat{x}) = J(\hat{x}) \quad \forall x \in S$, pues $\nabla J(\hat{x}) = 0$. □

1.3. Métodos de resolución

Recordemos que nuestro problema consiste en hallar un $\hat{x} \in S$ tal que $J(\hat{x}) = \min_{x \in S} J(x)$. Nuestros métodos de resolución buscarán disminuir el funcional coste. Resulta interesante destacar que los métodos que presentamos supondrán una regularidad mínima sobre J, $J \in \mathscr{C}^1$, o incluso $J \in \mathscr{C}^2$. Bajo estás condiciones si existe ese $\hat{x}$ tenemos que $\nabla J(\hat{x}) = 0$. Así $\nabla J(\hat{x}) = 0$ se usará como test de parada.

Las siguientes definiciones son interesantes para la implementación de los métodos.

Definición 1.27. Sea S abierto, $J : S \to \mathbb{R}$ diferenciable, $x \in S$, $u \in \mathbb{R}^n$ y α el ángulo que forman $\nabla J(x)$ y u.

$$\frac{\partial J}{\partial u}(x) := \langle \nabla J(x), u \rangle = ||\nabla J(x)||\ ||u|| \cos(\alpha)$$

es la *derivada direccional de J en el punto x en la dirección u.*

Observación 1.28. Nótese que:

- $\frac{\partial J}{\partial u}(x)$ es máximo cuando $\alpha = 0$, es decir, $\nabla J(x)$ indica la dirección de máximo crecimiento.
- $\frac{\partial J}{\partial u}(x)$ es mínimo cuando $\alpha = \pi$, es decir, $-\nabla J(x)$ indica la dirección de mínimo crecimiento o máximo decrecimiento.

Definición 1.29. Sea S abierto, $J : S \to \mathbb{R}$ diferenciable y $x \in S$. Dado $d \in \mathbb{R}^n$ diremos que J *decrece desde* x *en la dirección* d si $\langle \nabla J(x), d \rangle < 0$. En tal caso, d se llama *dirección de descenso para* J *desde el punto* x.

Vamos a ver ahora el uso de métodos iterativos para encontrar el mínimo en un problema de optimización. Nos centraremos en los llamados métodos de descenso. La idea es, dado un punto inicial, crear una sucesión haciendo uso de un método iterativo. Queremos con esto llegar a la solución del problema, construyendo así una sucesión que converge al mínimo.

Para finalizar con las iteraciones tendrá que cumplirse un cierto criterio de convergencia. También se dará un número máximo de iteraciones el cual si es alcanzado se dirá que el método usado no converge. Uno de los criterios que vamos a usar y quizás el más intuitivo será $||\nabla J(x_k)|| \leq \delta$, donde δ será lo que denominaremos *parámetro de tolerancia*. Que este criterio se cumpla nos indica que el gradiente tiende a cero y que la sucesión iterativa converge a un punto estacionario.

Existen distintos métodos iterativos de optimización. Muchos de ellos se engloban dentro de los métodos de descenso denominados así debido a que el valor de la función objetivo en cada iterante es menor que su valor en el anterior. Veamos el algoritmo básico de optimización esquematizado

Algoritmo 1.30. *En primer lugar se fijan un número máximo de iteraciones nitmax y los parámetros de tolerancia* $\epsilon > 0$ *y* $\delta > 0$ *(pequeños) que se usarán como criterios de parada.*

Paso 0 (Inicialización) Se proporciona una semilla $x_0 \in \mathbb{R}^n$*, si es posible cerca de* $\hat{x}$*,* $k := 0$*.*

Paso 1 (Dirección de descenso) Se elige una dirección de descenso d_k *para* J *desde* x_k*, se debe cumplir* $\langle \nabla J(x_k), d_k \rangle < 0$*.*

Paso 2 (Cálculo de paso) Se calcula $\alpha_k \in (0, +\infty)$ *paso tal que* $J(x_k + \alpha_k d_k) < J(x_k)$*.*

Paso 3 (Bucle) Se actualiza $x_{k+1} = x_k + \alpha_k d_k$*,* $k := k + 1$*.*

Paso 4 (Criterios de convergencia) Se comprueban que $\frac{||x_{k+1}-x_k||}{1+||x_{k+1}||} \leq \epsilon$ *y* $||\nabla J(x_{k+1})|| \leq \delta$ *si se cumple se terminan las iteraciones y* x_{k+1} *se considera una buena aproximación de* $\hat{x}$*. En otro caso se vuelve al Paso 1.*

Si el número de iteraciones alcanza nitmax sin que se cumplan los criterios de convergencia concluimos que el método no converge.

Veremos ahora dos resultados que nos resultará de utilidad para demostraciones posteriores (Teorema 1.38 y Teorema 1.47 respectivamente) en relación con el algoritmo anterior.

Teorema 1.31. *Sea* $\alpha_k > 0$ *la solución de* $J(x_k + \alpha_k d_k) = \min_{\alpha \in [0,+\infty)} J(x_k + \alpha d_k)$*, donde* d_k *es una dirección de descenso. Sea* $M > 0$ *tal que* $||HJ(x_k + \alpha d_k)|| \leq M \ \forall \alpha > 0$*. Entonces*

$$J(x_k) - J(x_k + \alpha_k d_k) \geq \frac{1}{2M}||\nabla J(x_k)||^2 \cos^2\langle d_k, -\nabla J(x_k)\rangle.$$

Demostración. Por hipótesis tenemos que

$$J(x_k + \alpha d_k) \leq J(x_k) + \alpha d_k^T \nabla J(x_k) + \frac{\alpha^2}{2} M||d_k||^2, \ \forall \alpha > 0.$$

Dado $\hat{\alpha} = -d_k^T \nabla J(x_k)/(M||d_k||^2)$; tenemos que

$$\begin{aligned} J(x_k) - J(x_k + \alpha_k d_k) &\geq J(x_k) - J(x_k + \hat{\alpha} d_k) \\ &\geq -\hat{\alpha} d_k^T \nabla J(x_k) - \frac{\hat{\alpha}^2}{2} M||d_k||^2 \\ &= \frac{1}{2}\frac{(d_k^T \nabla J(x_k))^2}{M||d_k||^2} \\ &= \frac{1}{2M}||\nabla J(x_k)||^2 \frac{(d_k^T \nabla J(x_k))^2}{||d_k||^2||\nabla J(x_k)||^2} \\ &= \frac{1}{2M}||\nabla J(x_k)||^2 \cos^2\langle d_k, -\nabla J(x_k)\rangle. \end{aligned}$$

□

Teorema 1.32. *Sea* $\nabla J(x)$ *uniformemente continua en el conjunto de nivel* $L(x_0) = \{x \in \mathbb{R}^n | J(x) < J(x_0)\}$*. Sea* θ_k *el ángulo que forman* $\nabla J(x_k)$ *y la dirección* d_k *elegida en el algoritmo 1.30 cumpliendo* $\theta_k \leq \frac{\pi}{2} - \mu$*, para algún* $\mu > 0$*. Entonces* $\nabla J(x_k) = 0$ *para un cierto* k*; o bien* $J(x_k) \to -\infty$*; o* $\nabla J(x_k) \to 0$*.*

Demostración. Supongamos que para todo k, $\nabla J(x_k) \neq 0$ y $J(x_k)$ está acotada inferiormente. Como $\{J(x_k)\}$ es monótona decreciente, existe su límite. Además

$$J(x_k) - J(x_{k+1}) \to 0. \tag{1.1}$$

Supongamos que no se cumple $\nabla J(x_k) \to 0$. Entonces existe $\epsilon > 0$ y un subconjunto K, tal que $||\nabla J(x_k)|| \geq \epsilon \; \forall k \in K$. Además

$$-\nabla J(x_k)^T d_k / ||\nabla J(x_k)|| \cos\theta_k \geq \epsilon \sin\mu = \epsilon_1 \tag{1.2}$$

Notese además que

$$\begin{aligned} J(x_k + \alpha d_k) &= J(x_k) + \alpha \nabla J(\xi_k)^T d_k \\ &= J(x_k) + \alpha \nabla J(x_k)^T d_k + \alpha [\nabla J(\xi_k) - \nabla J(x_k)]^T d_k \\ &\leq J(x_k) + \alpha ||d_k|| \left(\frac{\nabla J(x_k)^T d_k}{||d_k||} + ||\nabla J(\xi_k) - \nabla J(x_k)|| \right), \end{aligned} \tag{1.3}$$

donde ξ_k se encuentra entre x_k y $x_k + \alpha d_K$. Como $\nabla J(x)$ es uniformemente continuo en el conjunto de nivel $L(x_0)$, existe $\bar{\alpha}$ tal que cuando $0 \leq \alpha ||d_k|| \leq \bar{\alpha}$, tenemos

$$||\nabla J(\xi_k) - \nabla J(x_k)|| \leq \frac{1}{2}\epsilon_1 \tag{1.4}$$

Por (1.2)-(1.4) tenemos

$$J\left(x_k + \bar{\alpha}\frac{d_k}{||d_k||}\right) \leq J(x_k) + \bar{\alpha}\left(\frac{\nabla J(x_k)^T d_k}{||d_k||} + \frac{1}{2}\epsilon_1\right) \leq J(x_k) - \frac{1}{2}\bar{\alpha}\epsilon_1.$$

Entonces $J(x_{k+1}) \leq J\left(x_k + \bar{\alpha}\frac{d_k}{||d_k||}\right) \leq J(x_k) - \frac{1}{2}\bar{\alpha}\epsilon_1$, lo cual contradice (1.1). Por lo tanto $\nabla J(x_k) \to 0$. □

1.3.1. Selección de paso

En esta sección vamos a ver distintas reglas que nos permitirán calcular el paso a usar en los algoritmos de optimización.

Nos va a resultar útil considerar la función $j : (0, +\infty) \to \mathbb{R}$ definida por

$$j(\alpha) = J(x_k + \alpha d_k).$$

Para que α sea adecuado tendrá que garantizar descenso, es decir, debería cumplirse $j(\alpha) < j(0)$.

Observación 1.33. Si J es diferenciable también lo será $j(.)$ y $j'(\alpha) = d_k^t \nabla J(x_k + \alpha d_k)$, entonces $j'(0) = d_k^t \nabla J(x_k) < 0$.

Si J es dos veces diferenciable también lo será $j(.)$ y $j''(\alpha) = d_k^t H J(x_k + \alpha d_k) d_k$.

Omitiremos los subíndices k para simplificar la notación.

Búsqueda lineal exacta

Buscamos el paso óptimo lo cual se trata de elegir $\hat{\alpha}$ de forma que $j(\hat{\alpha}) = \min_{\alpha \in (0,+\infty)} j(\alpha)$. Esta búsqueda del paso se denomina búsqueda lineal exacta u óptima. No siempre es posible encontrar el paso óptimo de manera exacta. En estos casos se recurre a búsquedas inexactas que veremos más adelante.

Definición 1.34. Sea $j : \mathbb{R} \to \mathbb{R}$, $\hat{\alpha} \in [0, +\infty)$, y $j(\hat{\alpha}) = \min_{\alpha \in [0,+\infty)} j(\alpha)$. Si existe $(a, b) \subset [0, +\infty)$ tal que $\hat{\alpha} \in (a, b)$, entonces (a, b) se denomina *intervalo de búsqueda.*

La situación ideal sería que exista y sea único dicho mínimo en el intervalo $(0, +\infty)$ y que esté localizado en un intervalo de búsqueda $(a, b) \subset (0, +\infty)$.

Dándose la situación ideal, para una función unimodal (la definición la podemos ver en el capítulo 1, teorema 1.3.14, página 44 de [4]), podremos utilizar un método directo basado en la eliminación de regiones. Se basa en ir reduciendo el intervalo de búsqueda eliminando subintervalos donde seguro no se encuentra el mínimo. Para ello se escogen dos puntos en el interior del intervalo y mediante el estudio del valor de la función en esos puntos se puede determinar en cual de los subintervalos determinados por esos puntos no está el mínimo. Para más detalles sobre esto se podría ver el capítulo 2, sección 2.3, página 84 de [4].

En el caso de que $j \in \mathscr{C}^1([a, b])$ podríamos obtener $\hat{\alpha}$ resolviendo la ecuación $j'(\alpha) = 0$ por el método de bisección o dicotomía. Si $j \in \mathscr{C}^2([a, b])$ será generalmente mejor usar Newton para resolver la ecuación. Sin embargo estos métodos no garantizan el descenso.

Reglas aproximadas para la selección de paso

Usualmente la búsqueda lineal exacta resulta costosa a nivel computacional. Sin embargo siempre que podamos asegurar que la función objetivo tiene un descenso suficiente, se pueden evitar estos métodos reduciendo los costes computacionales. Hablaremos ahora de búsqueda lineal inexacta y nos centraremos en las reglas de Armijo, Goldstein y Wolfe-Powell. Estas búsquedas pretenden dar un paso que garantice el descenso. Las dos últimas además intentan conseguir que el paso no sea pequeño.

La regla de Armijo usa la recta que pasa por $(0, j(0))$ de pendiente $\rho j'(0)$, con $\rho \in (0, \frac{1}{2})$, que denotaremos por $r(\alpha)$. Podemos resumirla diciendo que escoge el paso igual a $\beta^m \tau$, con $\beta \in (0, 1)$ y siendo m el menor entero no negativo tal que $j(\beta^m \tau) \leq j(0) + \rho j'(0) \beta^m \tau = r(\beta^m \tau)$.

Algoritmo 1.35. *(Regla de Armijo)*

Paso 0 Se eligen $\tau > 0$, $\rho \in (0, \frac{1}{2})$ *y* $\beta \in (0, 1)$ *y se fija* $i := 0$.

Paso 1 *Se calculará* $r(\beta^i\tau) = j(0) + \rho j'(0)\beta^i\tau$.

Paso 2 *Se comprueba si* $j(\beta^i\tau) \leq r(\beta^i\tau)$. *En caso afirmativo* $\alpha_k = \beta^i\tau$ *y terminamos*

Paso 3 *Actualizamos* $i := i + 1$ *y se vuelve al Paso 1.*

Hablando ahora de la regla de Goldstein se elige $\rho \in (0, \frac{1}{2})$. Sean $r_1(\alpha) = j(0) + \rho j'(0)\alpha$ y $r_2(\alpha) = j(0) + (1-\rho)j'(0)\alpha$ esta regla elige α_k tal que $r_2(\alpha_k) \leq j(\alpha_k) \leq r_1(\alpha_k)$. Esta regla, a diferencia de la de Armijo que podría escoger pasos arbitrariamente pequeños, rechaza los pasos demasiado pequeños.

Algoritmo 1.36. *(Regla de Goldstein)*

Paso 0 *Se elige* $\alpha_{max} > 0$. $a_0 = 0$, $b_0 = \alpha_{max}$, $\alpha_0 = \frac{a_0+b_0}{2}$, $q = 0$;

Paso 1 *Se comprueba si* $j(\alpha_q) \leq r_1(\alpha_q)$.

En caso afirmativo Paso 2.

En caso negativo $a_{q+1} = a_q$, $b_{q+1} = \alpha_q$ *y vamos al Paso 3.*

Paso 2 *Se comprueba si* $r_2(\alpha_q) \leq j(\alpha_q)$.

En caso afirmativo $\alpha_k = \alpha_q$ *y terminamos.*

En caso negativo $a_{q+1} = \alpha_q$, $b_{q+1} = b_q$. *y vamos al Paso 3.*

Paso 3 $\alpha_{q+1} = \frac{a_{q+1}+b_{q+1}}{2}$, $q = q + 1$ *y volvemos al Paso 1.*

La regla de Wolfe-Powell resulta muy similar a la de Goldstein por lo cual la definiremos centrándonos en las diferencias con esta. Se elige $\sigma \in (\rho, 1)$ y se cambia en la regla de Goldstein $r_2(\alpha_k) \leq j(\alpha_k)$ por $j'(\alpha_k) \geq \sigma j'(0)$.

1.3.2. Selección de dirección

Veremos a continuación diferentes métodos de resolución que básicamente se diferenciaran en la forma en la que escogen la dirección. Tendremos el método del gradiente, el método de Newton, el gradiente conjugado y para finalizar los métodos cuasi-Newton.

Método del gradiente

Se llama método del gradiente al método que toma como dirección de descenso

$$d_k = -\nabla J(x_k).$$

Cuando además el paso α_k es óptimo se puede llamar método de máximo descenso o método del gradiente con paso óptimo. Si en cambio escogemos un paso conveniente para todas las iteraciones lo llamaremos método del gradiente con paso fijo. En lo que sigue hablaremos del gradiente con paso óptimo por ser el que escoge el paso fijo una simplificación de este.

Para simplificar la notación consideramos $g_k = \nabla J(x_k)$.

Algoritmo 1.37.

Paso 0 *Fijamos la tolerancia $\delta \in (0,1)$ cerca del cero. Damos un punto inicial $x_0 \in \mathbb{R}^n$ y inicializamos $k := 0$.*

Paso 1 *Si $||g_k|| \leq \delta$, obtenemos $\hat{x} = x_k$; en otro caso $d_k = -g_k$.*

Paso 2 *Buscamos el paso α_k, tal que $J(x_k + \alpha_k d_k) = \min_{\alpha \in [0,+\infty)} J(x_k + \alpha d_k)$.*

Paso 3 *Calculamos $x_{k+1} = x_k + \alpha_k d_k$, $k := k+1$ y volvemos al Paso 1.*

Veremos ahora un teorema para la convergencia global del método del gradiente.

Teorema 1.38. *Sea $J \in \mathscr{C}^2(\mathbb{R}^n)$ y $||HJ(x)|| \leq M$, $M > 0$ constante. Dados un punto x_0 y una tolerancia $\epsilon > 0$ iniciales, la secuencia generada por el método o bien termina en un número finito de iteraciones, o $\lim_{k\to\infty} J(x_k) = -\infty$, o $\lim_{k\to\infty} \nabla J(x_k) = 0$.*

Demostración. Consideramos el caso infinito. Teniendo en cuenta el Algoritmo 1.37 y el Teorema 1.31, tenemos $J(x_k) - J(x_{k+1}) \geq \frac{1}{2M}||\nabla J(x_k)||^2$.

Entonces

$$J(x_0) - J(x_k) = \sum_{i=0}^{k-1}[J(x_i) - J(x_{i+1})] \geq \frac{1}{2M}\sum_{i=0}^{k-1}||\nabla J(x_k)||^2.$$

Tomamos límites y tenemos que o bien $\lim_{k\to\infty} J(x_k) = -\infty$, o $\lim_{k\to\infty} \nabla J(x_k) = 0$. □

Observación 1.39 (Gradiente estocástico). Introducimos ahora una variante de método del gradiente para funcionales que involucran una suma de muchos términos individuales. Tendremos que dado $J(x) = \frac{1}{n}\sum_{i=1}^n h_i(x)$ entonces $\nabla J(x) = \frac{1}{n}\sum_{i=1}^n \nabla h_i(x)$. Para valores de n muy grandes podría ser muy costoso calcular este gradiente en cada iteración. Es por eso que introducimos esta variante que reduce considerablemente el coste computacional.

Veremos la versión más simple del gradiente estocástico en la cual en cada paso se usa un elemento escogido de manera aleatoria para representar al conjunto completo. Lo que se hará es en cada iteración del método en lugar de usar $-\nabla J(x)$ como dirección de

descenso se escogerá un entero i de manera uniformemente aleatoria en $\{1, 2, \ldots, n\}$ y se usará $-\nabla h_i(x)$.

A medida que avanzan las iteraciones del método se usan nuevos elementos así que tenemos la esperanza de que esta solución merezca la pena por su reducción del coste en cada iteración. En este método será preferible el uso de un paso fijo pues al cambiar la media por un único valor no tenemos asegurado que se reduzca el valor del funcional.

Esto se da en el caso de que tengamos un funcional coste del tipo mínimos cuadrados con muchos datos. En este caso todos los términos serán de la misma naturaleza (h_i independiente de i, x_i datos y $h_i = h(x - x_i)$). Así se remplazaría el verdadero gradiente (media de los gradientes) por uno de ellos. Veremos el uso de este método en el entrenamiento de la red neuronal (sección 3.3.2).

Método de Newton

Supongamos $J \in \mathscr{C}^2(S)$. En principio este método no es como los descritos en el algoritmo 1.30. Afronta de forma directa la resolución del sistema $\nabla J(x) = 0$ mediante Newton.

Observación 1.40. Recordemos que para un función $f : \mathbb{R} \to \mathbb{R}$, buscando x tal que $f(x) = 0$ las iteraciones con el método de Newton serían dado $x_0 \in \mathbb{R}$,

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k \geq 0.$$

Para $F : \mathbb{R}^n \to \mathbb{R}^n$ buscando x tal que $F(x) = 0$ dado $x_0 \in \mathbb{R}^n$, tenemos

$$x_{k+1} = x_k - (DF(x_k)^{-1}(F(x_k)), \quad k \geq 0.$$

De esta última ecuación obtenemos $DF(x_k)(x_{k+1} - x_k) = -F(x_k)$.

La implementación del método se hará resolviendo el sistema lineal $DF(x_k)w = -F(x_k)$. Luego $x_{k+1} = x_k + w$.

Aplicando el método de Newton para resolver el sistema $\nabla J(x) = 0$ y dado $x_0 \in \mathbb{R}^n$ tenemos

$$x_{k+1} = x_k - (HJ(x_k))^{-1}(F(x_k)), \quad k \geq 0.$$

Para la implementación se resuelve $HJ(x_k)w = -\nabla J(x_k)$ y $x_{k+1} = x_k + w$.

Observación 1.41. La dirección del método de Newton $d_k = -(HJ(x_k))^{-1}(\nabla J(x_k))$ es una dirección de descenso pues se cumple que

$$\langle d_k, \nabla J(x_k)\rangle = (d_k)^T \nabla J(x_k) = -(\nabla J(x_k))^T \left[(HJ(x_k))^{-1}\right]^T \nabla J(x_k) < 0,$$

si $HJ(x_k)$ es definida positiva.

Para simplificar la notación del algoritmo identificamos $g_k := \nabla J(x_k)$ y $H_k = HJ(x_k)$.

Algoritmo 1.42.

Paso 0 *Damos* $x_0 \in \mathbb{R}^n$, $\epsilon > 0$ *e inicializamos k:=0;*

Paso 1 *Si* $||g_k|| \leq \epsilon$, $\hat{x} = x_k$ *y terminamos;*

Paso 2 *Obtenemos* d_k *resolviendo* $H_k d_k = -g_k$*;*

Paso 3 $x_{k+1} = x_k + d_k$*;*

Paso 4 $k := k + 1$ *y volvemos al Paso 1.*

Observación 1.43. Para una función definida positiva cuadrática, el método de Newton obtiene el resultado en una iteración. En general ni siquiera podemos garantizar que lo haga en un número finito de ellas. Sin embargo, cuando la función objetivo se asemeja a una función cuadrática cerca del mínimo, si el punto inicial está lo suficientemente cerca de él, el método converge de manera rápida.

En el siguiente teorema se muestra bajo qé condiciones se da la convergencia local del método de Newton y bajo cuales esa convergencia es cuadrática.

Teorema 1.44. *Sean* $J : S \subset \mathbb{R}^n \to \mathbb{R}$, S *conjunto abierto,* $J \in \mathscr{C}^2(S)$. *Sean* $\hat{x} \in S$ *la solución del problema de minimización y* x_k *lo suficientemente cerca de* $\hat{x}$ *tal que* $\nabla J(x_k) = 0$ *y* $HJ(x_k)$ *es definida positiva.Si además* HJ *es Lipschitziana, la sucesión* $\{x_k\}$ *generada por el método de Newton converge a* $\hat{x}$ *con convergencia cuadrática.*

Demostración. Por ser HJ Lipschitziana tenemos que $|HJ_{ij}(x) - HJ_{ij}(y)| \leq \gamma||x - y||$, para algún γ y para todos los i, j, donde $HJ_{ij}(x)$ es el elemento (i, j) de $HJ(x)$, entonces la actualización del método de Newton está bien definida.

Usando la fórmula de Taylor tenemos

$$0 = \nabla J(\hat{x}) = \nabla J(x_k) - HJ(x_k)(x_k - \hat{x}) + O(||x_k - \hat{x}||^2).$$

Como $J \in \mathscr{C}^2(S)$, x_k está lo suficientemente cerca de $\hat{x}$, y $HJ(x_k)$ es definida positiva. Entonces la k-ésima iteración del método de Newton existe. Multiplicando por $HJ(x_k)^{-1}$ tenemos $0 = HJ(x_k)^{-1}\nabla J(x_k) - (x_k - \hat{x}) + O(||x_k - \hat{x}||^2)$. Usando la notación del algoritmo 1.42 tenemos que esto es igual a $-s_k - h_k + O(||h_k||^2) = -h_{k+1} + O(||h_k||^2)$.

Por la definición de $O(.)$, existe una constante β tal que $||h_{k+1}|| \leq \beta||h_k||^2$. Si $x_k \in \Omega = \{x| \, ||h|| \leq \gamma/\beta, h = x - \hat{x}, \gamma \in (0, 1)\}$, entonces $||h_{k+1}|| \leq \gamma||h_k|| \leq \gamma^2/\beta < \gamma/\beta$. De modo que $x_{k+1} \in \Omega$. Por inducción en k, las iteraciones del método de Newton están bien definidas para todo k, y $||h_k|| \to 0$ con $k \to \infty$. Por lo tanto las iteraciones convergen. Además, teniendo en cuenta $||h_{k+1}|| \leq \beta||h_k||^2$ la convergencia es cuadrática. □

Habría que tener cuidado con este método pues no garantiza el descenso. Vemos que el método de Newton es un método local y tenemos problemas cuando el punto inicial está lejos de la solución. Podremos usar el método de Newton con la búsqueda lineal del paso para mitigar este efecto.

Algoritmo 1.45.

Paso 0 *Damos $x_0 \in \mathbb{R}^n$, $\epsilon > 0$ y inicializamos $k := 0$.*

Paso 1 *Calculamos g_k. Si $||g_k|| \leq \epsilon$, $\hat{x} = x_k$ y terminamos;*

Paso 2 *Resolvemos $H_k d_k = -g_k$ para obtener d_k;*

Paso 3 *Calculamos el paso, α_k, con alguno de los métodos de búsqueda lineal;*

Paso 4 *Actualizamos $x_{k+1} = x_k + \alpha_k d_k$, $k := k + 1$ y volvemos al Paso 1*

Teorema 1.46. *Sea $J \in \mathscr{C}^2(S)$ siendo $S \in \mathbb{R}^n$ un conjunto convexo no vacío. Supongamos que existe una constante $m > 0$ tal que $J(x)$ satisface*

$$u^T HJ(x)u \geq m||u||^2, \forall u \in \mathbb{R}^n,\ x \in L(x_0). \tag{1.5}$$

Entonces el conjunto de nivel $L(x_0) = \{x | J(x) \leq J(x_0)\}$ es un conjunto cerrado, convexo y acotado.

Demostración. Demostración en el capítulo 1, teorema 1.3.19, página 47 de [4]. □

Teorema 1.47. *Sea $J : \mathbb{R}^n \to \mathbb{R}$, $D \subset \mathbb{R}^n$ abierto convexo, $J \in \mathscr{C}^2(D)$. Teniendo que para cualquier $x_0 \in D$ existe una constante $m > 0$ tal que $J(x)$ satisface*

$$u^T HJ(x)u \geq m||u||^2, \forall u \in \mathbb{R}^n,\ x \in L(x_0). \tag{1.6}$$

Entonces la sucesión x_k generada por el algoritmo anterior satisface $g_k = 0$ para algún k cuando la sucesión es finita y converge al único mínimo $\hat{x}$ de J cuando es infinita.

Demostración. En primer lugar, por (1.6), tenemos que $J(x)$ es una función estrictamente convexa en $\mathbb{R}^n$, y por tanto su punto estacionario será el único mínimo global.

Además, por hipótesis, $L(x_0)$ es un conjunto cerrado convexo y acotado (Teorema 1.46). Como $\{J(x_k)\}$ es monótona decreciente, $\{x_k\} \subset L(x_0)$ y $\{x_k\}$ está acotada. Existe un punto de acumulación $\hat{x} \in L(x_0)$ con $x_k \to \hat{x}$, y $J(x_k) \to J(\hat{x})$. A mayores como $J \in \mathscr{C}^2(D)$, por el teorema 1.32, tenemos $g_k \to g(\hat{x}) = 0$. Finalmente, viendo que el punto estacionario es único, la sucesión $\{x_k\}$ converge a $\hat{x}$ que es el único mínimo. □

Gradiente conjugado

Veremos ahora el método del gradiente conjugado que presenta una convergencia más rápida que el gradiente y evita el calculo de hessianas que puede ser costoso ya que este método solo usa las derivadas de primer orden. Consideraremos para la presentación de estos métodos el caso cuadrático asumiendo que $J(x) = \frac{1}{2}x^T Ax + b^T x + c$.

Introducimos en primer lugar las direcciones conjugadas y los métodos que usan estas direcciones pues el método del gradiente conjugado es uno de ellos.

Definición 1.48. Sean A una matriz simétrica y definida positiva $n \times n$, $d_1, \ldots, d_m \in \mathbb{R}^n$ vectores distintos de cero, $m \leq n$. Si $d_i^T A d_j = 0$, $\forall i \neq j$, los vectores $d_1, \ldots, d_m$ se dicen *$A-$conjugadas* o *simplemente conjugadas.*

Observación 1.49. Si $A = mI$ ser $A-$conjugado equivale a ser ortogonal.

Algoritmo 1.50. *(Método de dirección conjugada general)*

Paso 0 *Damos x_0, $\epsilon > 0$, $k := 0$.*

Paso 1 *Calculamos $g_0 = g(x_0)$, d_0 tal que $d_0^T g_0 < 0$.*

Paso 2 *Si $||g_k|| \leq \epsilon$, $\hat{x} = x_k$ y terminamos.*

Paso 3 *Calculamos α_k tal que $J(x_k + \alpha_k d_k) = \min_{\alpha \in [0,+\infty)} J(x_k + \alpha d_k)$.*

Paso 4 *Construimos $x_{k+1} = x_k + \alpha_k d_k$.*

Paso 5 *Calculamos d_{k+1} con algún método de dirección conjugada, tal que $d_{k+1}^T A d_j = 0$, $j = 0, 1, \ldots, k$.*

Paso 6 *Actualizamos $k := k + 1$ y volvemos las Paso 2.*

Teorema 1.51. *Dada una función cuadrática con matriz Hessiana definida positiva el método de dirección conjugada converge en a lo sumo n iteraciones de búsqueda lineal exacta.*

Demostración. Demostración en el capítulo 4, teorema 4.1.3, página 176 de [4]. □

Veremos ahora el método del gradiente conjugado con una explicación de como calcular las direcciones conjugadas. En primer lugar nos centraremos en el caso de funcionales cuadráticos.

Tenemos $J(x) = \frac{1}{2}x^T Ax + b^T x + c$, donde A es una matriz simétrica definida positiva $n \times n$, $b \in \mathbb{R}^n$ y c un número real. Definimos $g(x) = Ax + b$ el gradiente de J en x. Entonces

con $d_0 = -g_0$ tenemos $x_1 = x_0 + \alpha_0 d_0$, donde α_0 es óptimo. Así tenemos $g_1^T d_0 = 0$. Ahora $d_1 = -g_1 + \beta_0 d_0$ y escogemos β_0 tal que $d_1^T A d_0 = 0$. Podemos calcular β_0 multiplicando la formula de d_1 por $d_0^T A$ y obtenemos $\beta_0 = \frac{g_1^T A d_0}{d_0^T A d_0} = \frac{g_1^T (g_1 - g_0)}{d_0^T (g_1 - g_0)} = \frac{g_1^T g_1}{g_0^T g_0}$ nos interesa esta equivalencia ya que tenemos una formulación solo involucrando gradientes. En general tendremos $\beta_k = \frac{{g_k+1}^T g_k+1}{g_k^T g_k}$ (fórmula de Fletcher-Reeves) y siendo α_k paso óptimo $\alpha_k = \frac{-g_k^T d_k}{d_k^T A d_K}$.

Algoritmo 1.52. *(Gradiente conjugado caso cuadrático)*

Paso 0 *Damos* $x_0 \in \mathbb{R}^n$.

Paso 1 *Inicializamos* $g_0 = Ax_0 + b$, $d_0 = -g_0$. $(d_0 = - \bigtriangledown J(x_0))$

Paso 2 *Si* $d_0 = 0$, $\hat{x} = x_0$ *y terminamos.*

Paso 3 *Calculamos el paso óptimo* $\alpha_k = \frac{-g_k^T d_k}{d_k^T A d_k}$.

Paso 4 *Actualizamos* $x_{k+1} = x_k + \alpha_k d_k$ *y* $g_{k+1} = Ax_{k+1} + b$.

Paso 5 *Si* $g_{k+1} = 0$, $\hat{x} = x_{k+1}$ *y terminamos.*

Paso 6 *Calculamos* $\beta_k = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$.

Paso 7 *Actualizamos* $d_{k+1} = -g_{k+1} + \beta_k d_k$ *y volvemos al Paso 3.*

Algoritmo 1.53. *(Gradiente conjugado caso general)*

Paso 0 *Damos* $x_0 \in \mathbb{R}^n$ *y* $\epsilon < 0$.

Paso 1 $g_0 = \nabla J(x_0)$, $d_0 = -g_0$.

Paso 2 *Si* $||g_0|| \leq \epsilon$ $\hat{x} = x_0$ *y terminamos.*

Paso 3 *Calculamos el paso óptimo o aproximado.*

Paso 4 *Actualizamos* $x_{k+1} = x_k + \alpha_k d_k$.

Paso 5 $g_{k+1} = \nabla J(x_{k+1})$ *si* $||g_{k+1}|| \leq \epsilon$ $\hat{x} = x_{k+1}$ *y terminamos.*

Paso 6 *Calculamos* $\beta_k = \frac{||\nabla J(x_{k+1})||^2}{||\nabla J(x_k)||^2}$.

Paso 7 *Actualizamos* $d_{k+1} = -g_{k+1} + \beta_k d_k$ *y volvemos al Paso 3.*

Métodos cuasi-Newton

El coste computacional de calcular las Hessianas puede ser muy alto al igual que puede ser difícil evaluarlas e incluso que no sean calculables analíticamente. Para solucionar estos problemas que se pueden presentar en el método de Newton usamos los cuasi-Newton. Esta familia de métodos solo usa la evaluación de las funciones y de los gradientes y en relación con la Hessianas lo que se hace es generar una sucesión de aproximaciones. Cabe resaltar que a pesar de realizar estas aproximaciones la velocidad de convergencia comparándola con la del método del gradiente se mejora notablemente.

Vamos a aproximar las Hessianas procurando que las aproximaciones B_k sean simétricas, definidas positivas y que la dirección $d_k = -B_k^{-1}g_k$ sea de descenso. Veremos ahora qué condición debe satisfacer la sucesión $\{B_k\}$.

Sea $J : S \to \mathbb{R}$, $S \subset \mathbb{R}^n$ abierto, $J \in \mathscr{C}^2(S)$. En virtud del teorema de Taylor sobre $J(x)$

$$J(x) \approx J(x_{k+1}) + (\nabla J(x_{k+1}))^T(x - x_{k+1}) + \frac{1}{2}(x - x_{k+1})^T HJ(x_{k+1})(x - x_{k+1}).$$

y sobre $F(x) = \nabla J(x)$

$$\nabla J(x) \approx \nabla J(x_{k+1}) + HJ(x_{k+1})(x - x_{k+1})$$

y evaluando en $x = x_k$,

$$(HJ(x_{k+1})^{-1}(\nabla J(x_{k+1}) - \nabla J(x_k)) \approx x_{k+1} - x_k.$$

Para simplificar la notación tomamos $s_k = x_{k+1} - x_k$ y $y_k = \nabla J(x_{k+1}) - \nabla J(x_k)$. Ahora tenemos $HJ(x_{k+1})^{-1}y_k = s_k$ y le pedimos a la aproximación de la inversa de la Hessiana H_{k+1} que cumpla $H_{k+1}y_k = s_k$ que se llama ecuación o condición cuasi-Newton.

Considerando la función

$$m_{k+1}(x) = J(x_{k+1}) + g_{k+1}^T(x - x_{k+1}) + \frac{1}{2}(x - x_{k+1})^T B_{k+1}(x - x_{k+1})$$

que satisface $m_{k+1}(x_{k+1}) = J(x_{k+1})$ y $\nabla m_{k+1}(x_{k+1}) = g_{k+1}$. Pedimos que satisfaga $\nabla m_{k+1}(x_k) = g_k$, así

$$g_k = g_{k+1} + B_{k+1}(x_k - x_{k+1}).$$

Tenemos entonces que se cumple la condición cuasi-Newton. Si multiplicamos por s_k^T tenemos $s_k^T B_{k+1}s_k = s_k^T y_k$. Lo cual significa que si $s_k^T y_k > 0$, la matriz B_{k+1} es definida positiva.

Este método usa la aproximación de la inversa de la hessiana, H_k, para calcular el descenso, d_k, como $-H_k g_k$. Se da un valor inicial de la aproximación que se actualiza

en cada iteración de manera que cumpla la condición cuasi-Newton. Tenemos entonces el siguiente algoritmo del método. Será un algoritmo con la estructura del de un método de descenso, con el descenso calculado como hemos visto y que ademas debe añadir el paso de actualización de la inversa de la hessiana.

Algoritmo 1.54.

Paso 0 *Damos* $x_0 \in \mathbb{R}^n$, $H_0 \in \mathbb{R}^{n\times n}$ *(simétrica y definida positiva, por ejemplo* I*),* $0 \leq \epsilon < 1$ *e inicializamos* $k := 0$.

Paso 1 *Si* $||g_k|| \leq \epsilon$ *y terminamos.*

Paso 2 *Calculamos* $d_k = -H_k g_k$.

Paso 3 *Buscamos el paso óptimo* $\alpha_k > 0$ *y actualizamos* $x_{k+1} = x_k + \alpha_k d_k$.

Paso 4 *Actualizamos* H_k *a* H_{k+1} *de manera que se cumple la condición cuasi-Newton.*

Paso 5 $k := k + 1$ *y volvemos al Paso 1.*

También se puede usar este algoritmo para encontrar B_k en lugar de H_k. En el Paso 2 se resuelve $B_k d_k = -g_k$ y en el resto del algoritmo cambiamos H_k por B_k.

Veremos ahora dos formas distintas de actualizar H_k.

Actualización DFP (Davidon-Fletcher-Powell)

H_{k+1} se forma añadiendo a H_k dos matrices simétricas, cada una de rango uno. Consideramos la actualización simétrica de rango dos $H_{k+1} = H_k + auu^T + bvv^T$, donde $u, v \in \mathbb{R}^n$, a y b escalares. Haciendo que se cumpla la condición cuasi-Newton $H_k y_k + auu^T y_k + bvv^T y_k = s_k$. u y v no están unicamente determinadas, pero escogeremos $u = s_k$, $v = H_k y_k$ por ser las opciones más intuitivas. Entonces tenemos $a = \frac{1}{u^T y_k} = \frac{1}{s_k^T y_k}$, $b = \frac{-1}{v^T y_k} = \frac{-1}{y_k^T H_k y_k}$. Por tanto,

$$H_{k+1} = H_k + \frac{s_k s_k^T}{s_k^T y_k} - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} \tag{1.7}$$

A la hora de la implementación del algoritmo la actualización en el Paso 4 se hace con la fórmula anterior teniendo en cuenta que $s_k = \alpha_k d_k$, $x_{k+1} = x_k + s_k$, $y_k = g_{k+1} - g_k$.

Teorema 1.55. *La actualización 1.7 mantiene las matrices definidas positivas si ,y solo si,* $s_k^T y_k > 0$.

Actualización BFGS(Broyden-Fletcher-Goldfard-Shanno)

Inicialmente haciendo el cambio en la actualización anterior de H_k por B_k y de y_k y s_k entre sí obtenemos

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}.$$

Sean $A \in \mathscr{M}_{n\times n}(\mathbb{R})$ invertible, $x, y \in \mathbb{R}^n$ y $y^T A^{-1} x \neq -1$. Aplicando dos veces la fórmula de Sherman-Morrison

$$(A + xy^t)^{-1} = A^{-1} - (1 + y^T Ax)^{-1} A^{-1} x y^T A_{-1}$$

tenemos

$$\begin{aligned} H_{k+1} &= H_k + \left(1 + \frac{y_k^T H_k y_k}{s_k^T y_k}\right) \frac{s_k s_k^T}{s_k^T y_k} - \frac{s_k y_k^T H_k + H_k y_k s_k^T}{s_k^T y_k} \\ &= H_k + \frac{(s_k - H_k y_k) s_k^T + s_k (s_k - H_k y_k)^T}{s_k^T y_k} - \frac{(s_k - H_k y_k)^T y_k}{(s_k^T y_k)^2} s_k s_k^T \\ &= \left(I - \frac{s_k y_k^t}{s_k^T y_k}\right) H_k \left(I - \frac{y_k s_k^T}{s_k^t y_k}\right) + \frac{s_k s_k^T}{s_k^T y_k} \end{aligned}$$

Capítulo 2

Optimización con restricciones

En este capitulo haremos una revisión de los problemas de optimización con restricciones, su formulación, condiciones de optimalidad y hablaremos de uno de sus métodos de resolución que lo reduce a un problema de optimización sin restricciones. Trataremos además el método de los multiplicadores de Lagrange que nos dará una nueva condición necesaria de optimalidad. En primer lugar vamos a centrarnos en el caso en el que solo nos encontramos con restriciones de igualdad para a continuación abordar el caso en el que tenemos también restricciones de desigualdad.

2.1. Restricciones de igualdad

Sean $J : \mathbb{R}^n \to \mathbb{R}$ la función objetivo y $c_i : \mathbb{R}^n \to \mathbb{R}, i = 1; \ldots, m$ las funciones de restricción. La formulación general de este tipo de problemas de optimización con restricciones será:

$$\begin{cases} \min\limits_{x \in \mathbb{R}^n} & J(x) \\ \text{sujeto a} & c_i(x) = 0,\ i = 1, \ldots, m \end{cases} \tag{2.1}$$

Definición 2.1. Un punto $x \in \mathbb{R}^n$ se dice *factible* si, y solo si, $c_i(x) = 0$ para $i = 1, \ldots, m$. El conjunto de todos los puntos factibles denotados por $S = \{x \in \mathbb{R}^n | c_i(x) = 0\ \forall i \in I\}$ siendo $I = \{1, \ldots, m\}$, se denomina *conjunto factible*.

Vamos a suponer S no vacío y reescribiremos el problema como $\min\limits_{x \in S} J(x)$. Daremos a continuación un par de teoremas que nos serán de utilidad más adelante.

Teorema 2.2. *Supongamos que $c_i(x) \geq 0\ \forall x \in \mathbb{R}^n\ \forall i \in I$, c_i convexa $\forall i \in I$ entonces S es convexo.*

Demostración. Sean $\alpha_1 \geq 0,\ \alpha_2 \geq 0$ tal que $\alpha_1 + \alpha_2 = 1$

$$[x \in S, y \in S] \Rightarrow c_i(x) = 0 = c_i(y) \Rightarrow 0 \leq c_i(\alpha_1 x + \alpha_2 y) \leq \alpha_1 c_i(x) + \alpha_2 c_i(y) = 0$$
$$\Rightarrow c_i(\alpha_1 x + \alpha_2 y) = 0 \Rightarrow \alpha_1 x + \alpha_2 y \in S$$

□

Teorema 2.3. *Sea c función no nula convexa, entonces c^2 es convexa.*

Demostración. Sean $\alpha_1 \geq 0,\ \alpha_2 \geq 0$ tal que $\alpha_1 + \alpha_2 = 1$

$$c(\alpha_1 x + \alpha_2 y) \leq \alpha_1 c(x) + \alpha_2 c(y)$$
$$\Rightarrow c^2(\alpha_1 x + \alpha_2 y) \leq (\alpha_1 c(x) + \alpha_2 c(y))^2 = \alpha_1^2 c^2(x) + 2\alpha_1\alpha_2 c(x)c(y) + \alpha_2^2 c^2(y)$$

Usando $(a-b)^2 \geq 0 \Leftrightarrow 2ab \leq a^2 + b^2$ siendo $a = c(x)$ y $b = c(y)$ tenemos

$$\alpha_1^2 c^2(x) + 2\alpha_1\alpha_2 c(x)c(y) + \alpha_2^2 c^2(y) \leq \alpha_1^2 c^2(x) + \alpha_1\alpha_2(c^2(x) + c^2(y)) + \alpha_2^2 c^2(y)$$
$$= (\alpha_1^2 + \alpha_1\alpha_2)c^2(x) + (\alpha_2^2 + \alpha_1\alpha_2)c^2(y) = \alpha_1 c^2(x) + \alpha_2 c^2(y)$$

□

Observación 2.4. En el teorema 2.2 tenemos condiciones sobre las restricciones que de cumplirse el conjunto factible sería convexo. En el teorema 2.3 se habla de la convexidad de las restricciones al cuadrado. Resulta interesante ver que cumpliendo las restricciones las condiciones del teorema 2.3 entonces las restricciones al cuadrado cumplirían las condiciones del teorema 2.2. Teniendo en cuenta además la proposición 1.14 estos dos teoremas serán aplicables después con la definición de la función $\psi(.)$ asociada al método de penalización (ver (2.2)).

2.1.1. Condiciones basadas en los multiplicadores de Lagrange

Proporcionamos ahora condiciones necesarias que deben cumplirse en el óptimo para los problemas de optimización con restricciones de igualdad.

Definición 2.5. Consideraremos $\lambda \in \mathbb{R}^m$, la función

$$\mathcal{L}(x, \lambda) = J(x) + \sum_{i=1}^{m} \lambda_i c_i(x).$$

se denomina el *Lagrangiano asociado al problema (2.1)* y llamaremos a los λ_i *multiplicadores de Lagrange.*

Observación 2.6. Vista la definición anterior está claro que $\forall x \in \mathbb{R}^n$ tal que $c_i(x) = 0$ $\forall i \in I$ tenemos que $\mathcal{L}(x, \lambda) = J(x)$.

Teorema 2.7. *Será condición necesaria para que $\hat{x}$ sea mínimo del problema (2.1) que las derivadas parciales con respecto a todas las variables del Lagrangiano sean cero, $\nabla\mathcal{L}(\hat{x}, \hat{\lambda}) = 0$.*

Observación 2.8. Podemos observar que esto aplicado solo a la parciales con respecto a los multiplicadores de Lagrange es equivalente a que se cumplan las restricciones. Cumpliéndose entonces que $c_i(\hat{x}) = 0$ y $\partial_x J(\hat{x}) + \sum_i \hat{\lambda}_i \partial_x c_i(\hat{x}) = 0$ tendremos un sistema con el mismo número de ecuaciones que de incógnitas tal que si los c_i son lineales y los $\bigtriangledown c_i$ son linealmente independientes tendremos una única solución.

2.1.2. Método de penalización, caso de restricciones de igualdad

En este apartado daremos solo una breve descripción de este método. Nos centraremos más en él cuando a continuación estudiemos el caso más general.

Tomando la función

$$\psi(x) = \sum_{i=1}^{m} c_i^2(x) \tag{2.2}$$

podemos reescribir el conjunto S como $S = \{x \in \mathbb{R}^n | \psi(x) = 0\}$. Podemos también reescribir el problema de la siguiente manera $\min\limits_{x \in \mathbb{R}^n} J(x)$ sujeto a $\psi(x) = 0$.

Definición 2.9. Sea $J : S \subset \mathbb{R}^n \to \mathbb{R}$ con S no acotado, se dice que es coercitiva sobre S si $\lim\limits_{|x| \to \infty} J(x) = +\infty$.

Teorema 2.10. *Consideramos ahora $J : \mathbb{R}^n \to \mathbb{R}$ coercitiva y estrictamente convexa. Sea $S \subset \mathbb{R}^n$ no vacío, cerrado y convexo, $\psi : \mathbb{R}^n \to \mathbb{R}$ convexa tal que $\psi(x) \geq 0$ $\forall x \in \mathbb{R}^n$. Entonces $\forall \eta > 0$ $\exists \hat{x}_\eta \in \mathbb{R}^n$ tal que*

$$J_\eta(\hat{x}_\eta) = \min_{x \in \mathbb{R}^n} J_\eta(x)$$

donde

$$J_\eta(x) = J(x) + \eta\psi(x),$$

y además, $\lim\limits_{\eta \to \infty} \hat{x}_\eta = \hat{x}$, donde $\hat{x} \in S$ es el único tal que $J(\hat{x}) = \min\limits_{x \in S} J(x)$.

En este teorema 2.10 vemos que dado el problema de optimización con restricciones de igualdad (2.1) podemos aproximar su solución mediante la solución de un problema de optimización sin restricciones

$$\min_{x \in \mathbb{R}^n} J(x) + \eta\psi(x). \tag{2.3}$$

Si nos fijamos bien la manera de construir el funcional objetivo en (2.3), cuya solución aproximará la de (2.1), es añadirle al funcional objetivo de (2.1) un sumatorio de las funciones de restricción al cuadrado multiplicado por un número grande. Vemos entonces que lo que buscamos es obtener un funcional objetivo que sea igual al de (2.1) si se cumplen las condiciones de restricción y considerablemente mayor si no se cumplen. A una función construida de esta manera la llamaremos función de penalización.

Veremos ahora el algoritmo de este método para resolver (2.1)

Algoritmo 2.11.

Paso 0 *Damos* $x_0 \in \mathbb{R}^n$, $\eta_0 > 0$, $\rho \geq 0$ *e inicializamos* $k := 0$.

Paso 1 *Empezando en* x_k *buscamos la solución del problema* $\min\limits_{x\in\mathbb{R}^n} J(x) + \eta_k\psi(x)$ *usando alguno de los métodos vistos en el capítulo anterior. Obtenemos* x_{k+1}.

Paso 2 *Si* $||\psi(x_{k+1})|| \leq \rho$ *terminamos.*

Paso 3 *Actualizamos* $\eta_{k+1} = 10\eta_k$, $k := k + 1$ *y volvemos al Paso 1.*

2.2. Restricciones de igualdad y desigualdad

En este caso trabajamos con $J(x)$ la función objetivo y las funciones de restricción $c_i(x)$ $(i = 1, \ldots, m)$ regulares, de valor real evaluadas en $\mathbb{R}^n$. Sean m_e y m enteros no negativos tales que $0 \leq m_e \leq m$ tenemos la formulación general

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} & J(x) \\ \text{sujeto a} & c_i(x) = 0, \quad i = 1, \ldots, m_e; \\ & c_i(x) \geq 0, \quad i = m_e + 1, \ldots, m. \end{cases}$$

Simplificamos la notación considerando los conjuntos

$$E = \{1, \ldots, m_e\}$$

$$I = \{m_e + 1, \ldots, m\}$$

conjuntos de indices de las restricciones de igualdad y de desigualdad respectivamente. El conjunto factible será así $S = \{x | c_i(x) = 0,\ i \in E; c_i(x) \geq 0,\ i \in I\}$ y podemos reescribir el problema como $\min\limits_{x\in S} J(x)$.

Definición 2.12. Sea $x \in S$, si $c_i(x) = 0$, $i \in I$, diremos que c_i es una *restricción activa en* x en caso contrario diremos que c_i en una *restricción inactiva en* x. Siendo $I(x) = \{i \in I | c_i(x) = 0\}$, $\mathscr{A}(x) = E \bigcup I(x)$ es el *conjunto de los índices de las restricciones activas en* x.

Supongamos que $\hat{x}$ es un mínimo local, si hay un índice $i_0 \in I$ tal que $c_{i_0}(\hat{x}) > 0$, entonces, si eliminamos la restricción i_0-ésima, $\hat{x}$ sigue siendo el mínimo local del problema asociado. Vemos entonces que las i_0-ésima restricción es *inactiva en* $\hat{x}$ y una motivación para denominarla de esta manera.

Sea $\mathscr{A}(\hat{x})$ el conjunto de indices de las restricciones activas en $\hat{x}$, con $\hat{x}$ mínimo del problema. Entonces será suficiente con resolver

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} & J(x) \\ \text{sujeto a} & c_i(x) = 0, \quad i \in \mathscr{A}(\hat{x}). \end{cases}$$

En general será más fácil resolver el problema de optimización con restricciones de igualdad que el original. El problema es que $\hat{x}$ no se conoce.

2.2.1. Condiciones de optimalidad de primer orden

Empezamos esta sección dando una serie de definiciones que nos resultarán útiles para estudiar después las condiciones de optimalidad. Nos centraremos en dar los enunciados de estas condiciones pasando por alto las demostraciones de las mismas haciendo referencia a dónde se pueden encontrar.

Definición 2.13. Sean $x \in S$, $0 \neq d \in \mathbb{R}^n$. Si existe $\delta > 0$ tal que $x + td \in S$, $\forall t \in [0, \delta]$, entonces d se dice *dirección factible de* S *en* x. El conjunto de todas las direcciones factibles de S en x es

$$DF(x, S) = \{d | \exists \delta > 0 \text{ tal que } x + td \in S,\ \forall t \in [0, \delta]\}.$$

Observación 2.14. Con restricciones de igualdad en muchos casos no hay direcciones factibles como por ejemplo en optimmización sobre una circunferencia.

Definición 2.15. Sea $x \in S$ y $d \in \mathbb{R}^n$. Si

$$d^T \nabla c_i(x) = 0, \quad i \in E$$

$$d^T \nabla c_i(x) \geq 0, \quad i \in I(x)$$

entonces d se dice *dirección factible linealizada de* S *en* x. El conjunto de todas las direcciones factibles linealizadas de S en x es

$$DFL(x, S) = \{d | d^T \nabla c_i(x) = 0,\ i \in E;\ d^T \nabla c_i(x) \geq 0,\ i \in I(x)\}.$$

Definición 2.16. Sea $x \in S$ y $d \in \mathbb{R}^n$. Si existen sucesiones $\{d_k\}$ y $\{\delta_k\}$, $\delta_k > 0$ tal que $x + \delta_k d_k \in S$, $\forall k$ y $d_k \to d$, $\delta_k \to 0$, entonces la dirección límite d de llama *dirección*

factible secuencial de S en x. El conjunto de todas las direcciones factibles secuenciales de S en x es

$$DFS(x,S) = \{d | \exists \{d_k\} \subset \mathbb{R}^n,\ \{\delta_k\} \subset \mathbb{R},\ zx + \delta_k d_k \in S, \forall k;\ d_k \to d,\ \delta_k \to 0\}.$$

Observación 2.17. Hacemos notar que en este caso $\{x_k\}$ con $x_k = x + \delta_k d_k$ es una sucesión de puntos que satisface

1. $x_k \neq x, \quad \forall k$;
2. $\lim\limits_{k\to\infty} x_k = x$;
3. $x_k \in S \quad \forall k$ suficientemente grande.

Si fijamos $\delta_k = ||x_k - x||$, entonces tenemos $d_k = \frac{x_k - x}{||x_k - x||} \to d$ y $x_k = x + \delta_k d_k$ es una sucesión de puntos factibles.

Lema 2.18. *Sea $x \in X$. Si todas las funciones de restricción son diferenciables en x, entonces $DF(x,S) \subseteq DFS(x,S) \subseteq DFL(x,S)$.*

Demostración. Demostración en el capítulo 8, lema 8.2.4, página 389 de [4]. □

Nos interesa introducir el *conjunto de direcciones de descenso en x*

$$\mathcal{D}(x) = \{d | d^T \nabla J(x) < 0\}$$

para a continuación describir las condiciones necesarias para la existencia de un mínimo local.

Teorema 2.19. *Sea $\hat{x} \in S$ un mínimo local. Si $J(x)$ y $c_i(x)$ $(i \in I)$ son diferenciables en $\hat{x}$, entonces $d^T \nabla J(\hat{x}) \geq 0$, $\forall d \in DFS(\hat{x}, S)$, es decir, $DFS(\hat{x}, S) \bigcap \mathcal{D}(\hat{x}) = \emptyset$.*

Demostración. Demostración en el capítulo 8, teorema 8.2.5, página 390 de [4]. □

Recuperamos ahora el Lagrangiano

$$\mathcal{L}(x, \lambda) = J(x) - \sum_{i=1}^{m} \lambda_i c_i(x)$$

donde a $\lambda = (\lambda_1, \ldots, \lambda_m)^T \in \mathbb{R}^m$ será vector de multiplicadores de Lagrange.

Lema 2.20 (Lema de Farkas)**.** *El conjunto*

$$\left\{ d \left| \begin{array}{ll} d^T \nabla c_i(x) = 0, & i \in E; \\ d^T \nabla c_i(x) \geq 0, & i \in I(x), \\ d^T \nabla J(x) < 0 & \end{array} \right. \right\}$$

es un conjunto vacío si, y solo si, existe un número real λ_i, $i \in E$ y números reales no negativos $\lambda_i \geq 0$, $i \in I$ tales que

$$\nabla J(x) = \sum_{i \in E} \lambda_i \nabla c_i(x) + \sum_{i \in I(x)} \lambda_i \nabla c_i(x).$$

Teorema 2.21. *Sea $\hat{x}$ un mínimo local. Si se cumple la condición de cualificación*

$$DFS(\hat{x}, S) = DFL(\hat{x}, S),$$

entonces existen multiplicadores de Lagrange $\hat{\lambda}_i$ tales que se satisfacen las siguientes condiciones:

$$\nabla_x \mathcal{L}(\hat{x}, \hat{\lambda}) = \nabla J(\hat{x}) - \sum_{i=1}^{m} \hat{\lambda}_i \nabla c_i(\hat{x}) = 0 \qquad \textit{Condición de punto estacionario}$$

$$\left.\begin{array}{ll} c_i(\hat{x}) = 0, & \forall i \in E \\ c_i(\hat{x}) \geq 0, & \forall i \in I \end{array}\right\} \qquad \textit{Condiciones de punto factible}$$

$$\hat{\lambda}_i \geq 0, \quad \forall i \in I \qquad \textit{No negatividad de los multiplicadores}$$

$$\hat{\lambda}_i c_i(\hat{x}) = 0, \quad \forall i \in I \qquad \textit{Condición de complementaridad}$$

Estas condiciones se conocen como condiciones Karush-Kuhn-Tucker o KKT. Los puntos que cumplen estas condiciones se denominan puntos KKT.

Demostración. Como $\hat{x}$ es un mínimo local tiene que cumplir las condiciones de punto factible. Además sea $d \in DFS(\hat{x}, S)$, teniendo en cuenta el teorema 2.19, se tiene $d^T \nabla J(\hat{x}) \geq 0$. Por la condición de cualificación, tenemos $d \in DFL(\hat{x}, S)$. El sistema

$$\begin{array}{ll} d^T \nabla c_i(\hat{x}) = 0, & i \in E; \\ d^T \nabla c_i(\hat{x}) \geq 0, & i \in I(\hat{x}), \\ d^T \nabla J(\hat{x}) < 0 & \end{array}$$

no tendría solución. Por el lema de Farkas obtenemos inmediatamente

$$\nabla J(\hat{x}) = \sum_{i \in E} \hat{\lambda}_i \nabla c_i(\hat{x}) + \sum_{i \in I(\hat{x})} \hat{\lambda}_i \nabla c_i(\hat{x}),$$

donde $\hat{\lambda}_i \in \mathbb{R}$ para $i \in E$ y $\hat{\lambda}_i \geq 0$ para $i \in I(\hat{x})$. Fijando $\lambda_i = 0$ para $i \in I \backslash I(\hat{x})$, obtenemos $\nabla J(\hat{x}) = \sum_{i=1}^{m} \hat{\lambda}_i \nabla c_i(\hat{x})$, equivalente a la condición de punto estacionario. La condición de no negatividad de los multiplicadores es trivial.

Finalmente cuando $i \in I(\hat{x})$, $c_i(\hat{x}) = 0$ y $\hat{\lambda}_i \geq 0$ tenemos $\hat{\lambda}_i c_i(\hat{x}) = 0$; cuando $i \in I \backslash I(\hat{x})$, $c_i(\hat{x}) > 0$ pero $\hat{\lambda}_i = 0$, por lo tanto $\hat{\lambda}_i c_i(\hat{x}) = 0$. Con lo cual obtenemos $\hat{\lambda}_i c_i(\hat{x}) = 0$, $\forall i \in I$. □

Observación 2.22. En la condición de complementaridad lo que vemos es que $\hat{\lambda}_i$ y $c_i(\hat{x})$ no pueden ser los dos distintos de cero, es decir, los multiplicadores de Lagrange correspondientes a restricciones inactivas son cero. Las restricciones activas las definiremos como fuertes si el correspondiente multiplicador de Lagrange es mayor que cero y débiles si es cero.

Corolario 2.23. *Sea $\hat{x}$ un mínimo local. Si todas las restricciones $c_i(\hat{x})$, $i \in \mathscr{A}(\hat{x})$ son funciones lineales, entonces $\hat{x}$ es un punto KKT.*

La condición de cualificación que vemos en el inicio del teorema suele ser difícil de comprobar, a continuación damos algunos teoremas en los que vemos casos en los que resulta fácil hacerlo.

Teorema 2.24. *Sea x un punto factible y $\mathscr{A}(x)$ el conjunto de indices de las restricciones activas en x. Si $\nabla c_i(x)$, $i \in \mathscr{A}(x)$ son linealmente independientes, entonces la condición de cualificación se cumple.*

Demostración. Demostración en el capítulo 8, teorema 8.2.11, página 395 de [4]. □

Teorema 2.25. *Sea $\hat{x}$ un mínimo local. Si $\nabla c_i(\hat{x})$, $i \in \mathscr{A}(\hat{x})$, son linealmente independientes entonces hay multiplicadores de Lagrange $\hat{\lambda}_i$ $(i = 1, \ldots, m)$ tal que las condiciones de punto estacionario, de punto factible, de complementaridad y la no negatividad de los multiplicadores del teorema 2.21 se cumplen.*

Demostración. Véase el capítulo 8, teorema 8.2.12, página 396 de [4]. □

Teorema 2.26. *Sea $\hat{x}$ un mínimo local. Si se cumple la condición de regularidad*

$$DFS(\hat{x}, S) \bigcap \mathcal{D}(\hat{x}) = DFL(\hat{x}, S) \bigcap \mathcal{D}(\hat{x}),$$

entonces $\hat{x}$ es un punto KKT.

Demostración. Véase el capítulo 8, teorema 8.2.13, página 396 de [4]. □

Veamos ahora resultados sobre las condiciones suficientes de primer orden.

Teorema 2.27. *Sea $\hat{x} \in S$, $J(\hat{x})$ y $c_i(\hat{x})$ $(i = 1, \ldots, m)$ diferenciables en $\hat{x}$.*
Si $d^T \nabla J(\hat{x}) > 0$ $\forall d \in DFS(\hat{x}, S)$ no nulo, entonces $\hat{x}$ es un mínimo local estricto.

Demostración. Supongamos que $\hat{x}$ no es un mínimo local estricto, entonces existe $x_k \in S$ con $k = 1, 2, \ldots$ tal que $J(x_k) \leq J(\hat{x})$, y $x_k \to \hat{x}$, $x_k = \hat{x}$ para $k = 1, 2, \ldots$.

Sin pérdida de generalidad podemos asumir que $\frac{x_k-\hat{x}}{||x_k-\hat{x}||_2} \to d$. Fijamos $d_k = \frac{x_k-\hat{x}}{||x_k-\hat{x}||_2}$ y $\delta_k = ||x_k - \hat{x}||_2$. Por la definición de dirección factible secuencial, 2.16, tenemos que $d \in DFS(\hat{x}, S)$. Con lo anterior, tomando la ecuación

$$J(x_k) = J(\hat{x}) + (x_k - \hat{x})^T \nabla J(\hat{x}) + o(||x_k - \hat{x}||_2),$$

dividiendo entre $||x_k - \hat{x}||_2$ y haciendo el límite cuando $k \to \infty$, obtenemos $d^T \nabla J(\hat{x}) \leq 0$, lo cual, junto con $d \in DFS(\hat{x}, S)$ contradice la hipótesis del teorema. □

Corolario 2.28. *Sea $\hat{x} \in X$. Sean $J(\hat{x})$ y $c_i(\hat{x})$ $(i = 1, \ldots, m)$ diferenciables en $\hat{x}$. Si $d^T \nabla J(\hat{x}) > 0$ $\forall d \in DFL(\hat{x}, X)$ no nulo, entonces $\hat{x}$ es un mínimo local estricto.*

Veamos ahora condiciones de optimalidad en problemas de programación convexa. La definiremos como el problema de minimizar una función convexa en un conjunto convexo

$$\begin{cases} \text{mín} & J(x) \\ \text{sujeto a} & x \in S \end{cases} \tag{2.4}$$

donde $J(x)$ es convexa; $c_i(x)$, $i \in E$ funciones lineales y $c_i(x)$, $i \in I$ cóncavas; entonces el conjunto de restricciones $S = \{x | c_i(x) = 0,\ i \in E;\ \ c_i(x) \geq 0,\ i \in I\}$ es convexo.

Teorema 2.29. *Cada mínimo local de un problema convexo es también mínimo global y el conjunto S de mínimos globales es convexo.*

Demostración. Véase el capítulo 8, teorema 8.2.17, página 399 de [4]. □

Teorema 2.30. *Dado un punto $\hat{x}$ tal que sea un punto KKT. En el contexto de los problemas convexos este punto será el mínimo del problema (2.4).*

Demostración. Véase el capítulo 8, teorema 8.2.18, página 400 de [4]. □

Teorema 2.31. *Un problema de programación convexa (2.4) con función objetivo estrictamente convexa tiene un único mínimo.*

Demostración. Véase el capítulo 8, teorema 8.2.19, página 400 de [4]. □

2.2.2. Condiciones de optimalidad de segundo orden

Veremos en esta sección el papel que juegan las derivadas segundas de J y c_i con $i \in E \bigcup I$. Para las direcciones $d \in DFL(\hat{x}, S)$ tales que $d^T \nabla J(\hat{x}) = 0$, no podemos determinar, solo con las primeras derivadas, si al movernos sobre ellas aumentará o disminuirá la función objetivo. Necesitaremos ahora hipótesis más fuertes que en la sección anterior. Pediremos que tanto J como c_i, con $i \in E \bigcup I$, sean dos veces continuamente diferenciables.

Daremos al igual que antes alguna definición que nos resultará de interés para el estudio de las condiciones.

Definición 2.32. Sea x un punto KKT, y λ el vector de multiplicadores de Lagrange. Definimos el *conjunto de restricciones activas fuertes* como

$$I_+(x) = \{i | i \in I(x) \text{ con } \lambda_i > 0\}.$$

Definición 2.33. Sea x un punto KKT, y λ el vector de multiplicadores de Lagrange. Si d es una dirección factible linealizada y se cumple $\lambda_i d^T \nabla c_i(x) = 0 \ \forall i \in I(x)$, entonces d se dice *dirección de restricción linelizada nula*. El conjunto de todas estas direcciones será

$$G(x,\lambda) = \left\{ d \left| \begin{array}{ll} d \neq 0 & \\ d^T \nabla c_i(x) = 0, & i \in E \bigcup I_+(x) \\ d^T \nabla c_i(x) \geq 0, & i \in I \setminus I_+(x) \end{array} \right. \right\} = \left\{ d \left| \begin{array}{l} d \in DFL(x,\lambda) \\ d^T \nabla c_i(x) = 0, \ i \in I_+(x) \end{array} \right. \right\}$$

Observación 2.34. El conjunto $G(x,\lambda)$ contiene exactamente las direcciones $d \in DFL(x,S)$ en las cuales no esta claro usando los criterios que involucran solo a las primeras derivadas si el funcional objetivo aumenta o disminuye.

En primer lugar tenemos un teorema que nos da condiciones necesarias de segundo orden para la existencia de mínimo.

Teorema 2.35. *Sea $\hat{x}$ un mínimo local tal que se cumpla que los $\nabla c_i(x)$, $i \in \mathscr{A}(\hat{x})$ son linealmente independientes. Sea además $\hat{x}$ un punto KKT y $\hat{\lambda}$ el vector de multiplicadores de Lagrange correspondientes. Entonces*

$$d^T H_{xx} \mathcal{L}(\hat{x}, \hat{\lambda}) d \geq 0, \quad \forall d \in G(\hat{x}, \hat{\lambda}) \tag{2.5}$$

Demostración. Como $\hat{x}$ es un mínimo local, todas las sucesiones $\{x_k\}$ que tienden a $\hat{x}$ deben cumplir $J(x_k) \geq J(\hat{x})$ para k suficientemente grande. Lo que vamos a hacer es construir una sucesión de puntos factibles con dirección límite d y que muestre que $J(x_k) \geq J(\hat{x})$ implica que se cumple la condición 2.5.

Como $d \in G(\hat{x}, \hat{\lambda}) \subset DFL(\hat{x}, S)$, podemos escoger $\{t_k\}$ secuencia de escalares con $t_k > 0$ y construir una secuencia $\{x_k\}$ de punto factibles tendiendo a $\hat{x}$ tal que se cumple $\lim\limits_{k\to\infty} \frac{x_k - \hat{x}}{t_k} = d$, por tanto, $x_k = \hat{x} + t_k d + o(t_k)$. Dado que los $\bigtriangledown c_i$ son linealmente independientes tenemos que la matriz $C(\hat{x})$, con los gradientes de los c_i con $i \in \mathscr{A}(x)$, de rango m. Sea Z una matriz cuyas columnas son una base del espacio nulo de $C(\hat{x})$; es decir; $Z \in \mathbb{R}^{n \times (n-m)}$, $C(\hat{x})Z = 0$. Cogiendo $d \in DFL(\hat{x}, S)$ y suponiendo $\{t_k\}_{k=0}^{\infty}$ es una

secuencia de escalares positivos tales que $\lim_{k\to\infty} t_k = 0$. Definimos el sistema paramétrico de ecuaciones $R : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ como

$$R(x,t) = \begin{bmatrix} c(x) - tC(\hat{x})d \\ Z^T(x - \hat{x} - td) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.6}$$

y la solución $x = x_k$ de este sistema para $t = t_k > 0$ pequeños da una secuencia de puntos factibles que tiende a $\hat{x}$ y cumple $\lim_{k\to\infty} \frac{x_k - \hat{x}}{t_k} = d$.

Por (2.6) tenemos

$$c_i(x_k) = t_k \nabla c_i(\hat{x})^T d \quad \forall i \in \mathscr{A}(\hat{x}) \tag{2.7}$$

Teniendo ahora en cuenta como hemos definido el Lagrangiano; la definición de $G(x,\lambda)$, que nos da que con $d \in G(\hat{x},\hat{\lambda})$, $d^T \nabla J(\hat{x}) = \sum_{i \in E \bigcup I} \hat{\lambda}_i d^T \nabla c_i(\hat{x}) = 0$, y tambien (2.7) tenemos

$$\mathcal{L}(x_k,\hat{\lambda}) = \mathcal{L}(\hat{x},\hat{\lambda}) + (x_k - \hat{x})^T \nabla_x \mathcal{L}(\hat{x},\hat{\lambda}) + \frac{1}{2}(x_k - \hat{x})^T H_{xx}\mathcal{L}(\hat{x},\hat{\lambda})(x_k - \hat{x}) + o(||x_k - \hat{x}||^2). \tag{2.8}$$

Por la condición de complementaridad tenemos $\mathcal{L}(\hat{x},\hat{\lambda}) = J(\hat{x})$. Por la condición de punto estacionario $\nabla_x \mathcal{L}(\hat{x},\hat{\lambda}) = 0$. Por tanto usando $x_k - \hat{x} = t_k d + o(t_k)$ podemos reescribir (2.8) como

$$\mathcal{L}(x_k,\hat{\lambda}) = J(\hat{x}) + \frac{1}{2} t_k^2 d^t H_{xx}\mathcal{L}(\hat{x},\hat{\lambda}) + o(t_k^2) \tag{2.9}$$

y substituyendo tenemos

$$J(x_k) = J(\hat{x}) + \frac{1}{2} t_k^2 d^t H_{xx}\mathcal{L}(\hat{x},\hat{\lambda})d + o(t_k^2).$$

Si $d^T H_{xx}\mathcal{L}(\hat{x},\hat{\lambda})d < 0$, eso implica que $J(x_k) < J(\hat{x})$ para k suficientemente grande contradiciendo que $\hat{x}$ es mínimo local y por tanto se debe cumplir la condición (2.5). □

Veremos ahora las condiciones suficientes de segundo orden.

Teorema 2.36. *Sea $\hat{x}$ un punto KKT y $\hat{\lambda}$ el vector de multiplicadores de Lagrange asociado. Si $d^T H_{xx}\mathcal{L}(\hat{x},\hat{\lambda})d > 0$, $\forall d \in G(\hat{x},\hat{\lambda})$, entonces $\hat{x}$ es un mínimo local estricto.*

Demostración. Denotemos $\bar{G} = \{d \in G(\hat{x},\hat{\lambda}) / ||d|| = 1\}$ subconjunto compacto de $G(\hat{x},\hat{\lambda})$, entonces por hipótesis el mínimo de $d^T H_{xx}\mathcal{L}(\hat{x},\hat{\lambda})d$ en este conjunto es un número $\sigma > 0$. Por la construcción de $G(\hat{x},\hat{\lambda})$ tenemos que $\frac{d}{||d||} \in \bar{G}$ si , y solo si, $d \in G(\hat{x},\hat{\lambda})$, $d \neq 0$. Entonces

$$d^T H_{xx}\mathcal{L}(\hat{x},\hat{\lambda})d \geq \sigma ||d||^2 \quad \forall d \in G(\hat{x},\hat{\lambda}).$$

con $\sigma > 0$ definido como arriba.

Probaremos el resultado demostrando que para toda sucesión de puntos factibles $\{x_k\}$ tendiendo a $\hat{x}$ se tiene

$$J(x_k) \geq J(\hat{x}) + \frac{\sigma}{4}||x_k - \hat{x}||^2, \text{ para todo } k \text{ suficientemente grande.} \tag{2.10}$$

Supongamos que lo anterior no se cumple y existe una sucesión $\{x_k\}$ tendiendo a $\hat{x}$ tal que

$$J(x_k) < J(\hat{x}) + \frac{\sigma}{4}||x_k - \hat{x}||^2, \text{ para todo } k \text{ suficientemente grande.} \tag{2.11}$$

Tomando, si fuese necesario, una subsucesión podemos definir una dirección límite tal que $\lim_{k\to\infty} \frac{x_k - \hat{x}}{||x_k - \hat{x}||} = d$. Tenemos entonces $x_k = \hat{x} + t_k d + o(t_k)$ donde $t_k = ||x_k - \hat{x}||$. Tomando $i \in E$ y usando el teorema de Taylor tenemos

$$0 = \frac{1}{t_k} c_i(x_k) = \frac{1}{t_k}\left[c_i(\hat{x}) + t_k \nabla c_i(\hat{x})^T d + o(t_k)\right] = \nabla c_i(\hat{x})^T d + \frac{o(t_k)}{t_k}.$$

Haciendo el límite cuando $k \to \infty$, el último término de esta expresión tiende a cero y tenemos $\nabla c_i(\hat{x})^T d = 0$. Para $i \in I(x)$, tenemos

$$0 \leq \frac{1}{t_k} c_i(x_k) = \frac{1}{t_k}\left[c_i(\hat{x}) + t_k \nabla c_i(\hat{x})^T d + o(t_k)\right] = \nabla c_i(\hat{x})^T d + \frac{o(t_k)}{t_k}$$

y por argumento análogos $\nabla c_i(\hat{x})^T d \geq 0$. Con lo cual $d \in DFL(\hat{x}, S)$. Por como definimos el Lagrangiano y dado que $\hat{\lambda}_i \geq 0$, $c_i(x_k) \geq 0$ para $i \in I$ y $c_i(x_k) = 0$ para $i \in E$ tenemos

$$\mathcal{L}(x_k, \hat{\lambda}) = J(x_k) - \sum_{i \in \mathscr{A}(x_k)} \hat{\lambda}_i c_i(x_k) \leq J(x_k), \tag{2.12}$$

y al igual que en la demostración anterior tenemos (2.9).

Si d no estuviese en $G(\hat{x}, \hat{\lambda})$, podemos encontrar un $j \in I(\hat{x})$ tal que se cumple

$$\hat{\lambda}_j \bigtriangledown c_j(\hat{x})^T d > 0, \tag{2.13}$$

mientras que para el resto de indices $i \in \mathscr{A}(\hat{x})$ tenemos

$$\hat{\lambda}_i \bigtriangledown c_i(\hat{x})^T d \geq 0. \tag{2.14}$$

Por el teorema de Taylor y $\lim_{k\to\infty} \frac{x_k - \hat{x}}{||x_k - \hat{x}||} = d$, para este j tenemos

$$\hat{\lambda}_j c_j(x_k) = \hat{\lambda}_j c_j(\hat{x}) + \hat{\lambda}_j \nabla c_j(\hat{x})^T (x_k - \hat{x}) + o(||x_k - \hat{x}||) = ||x_k - \hat{x}|| \hat{\lambda}_j \nabla c_j(\hat{x})^T d + o(||x_k - \hat{x}||).$$

Entonces por (2.12)

$$\begin{aligned} \mathcal{L}(x_k, \hat{\lambda}) =& J(x_k) - \sum_{i \in \mathscr{A}(\hat{x})} \hat{\lambda}_i c_i(x_k) \leq J(x_k) - \hat{\lambda}_j c_j(x_k) \\ \leq& J(x_k) - ||x_k - \hat{x}|| \hat{\lambda}_j \bigtriangledown c_j(\hat{x})^T d + o(||x_k - \hat{x}||) \end{aligned} \tag{2.15}$$

Por la estimación basada en el teorema de Taylor (2.9) tenemos $\mathcal{L}(x_k, \hat{\lambda}) = J(\hat{x}) + O(||x_k - \hat{x}||^2)$ y combinando con (2.15) obtenemos

$$J(x_k) \geq J(\hat{x}) + ||x_k - \hat{x}||\hat{\lambda}_j \bigtriangledown c_j(\hat{x})^T d + o(||x_k - \hat{x}||).$$

Esto sería incompatible con (2.11) si se cumpliese (2.13) por tanto tenemos que para toda sucesión de puntos factibles $\{x_k\}$ tendiendo a $\hat{x}$ debe satisfacer $J(x_k) \geq J(\hat{x}) + \frac{\sigma}{4}||x_k - \hat{x}||^2$, para todo k suficientemente grande. Por tanto $\hat{x}$ es un mínimo local estricto. □

En el libro [5] encontramos varios ejemplos en los cuales se comprueban las condiciones de segundo orden que resultan interesantes.

2.2.3. Método de penalización

Veremos ahora un estudio más amplio del método de penalización centrado en el caso que estamos viendo actualmente. En esta clase de métodos resolveremos en lugar del problema original (2.2) una serie de problemas sin restricciones que minimizan las funciones de penalización.

Función de penalización

La función de penalización, $P(x) = P(J(x), c(x))$, se construye a partir de la función objetivo y de las restricciones y es una función con propiedades de penalización. La propiedad de penalización pide que $P(x) = J(x)$ para todos los puntos factibles y $P(x)$ mucho mayor que $J(x)$ cuando se incumplan las restricciones.

Definimos la siguiente función para de alguna manera cuantificar los incumplimientos

$$c^*(x) = (c_1^*(x), \ldots, c_m^*(x))^T$$

$$\begin{aligned} c_i^*(x) &= c_i(x), && i \in E; \\ c_i^*(x) &= \text{mín}\{c_i(x), 0\}, && i \in I. \end{aligned}$$

Para cada restricción la correspondiente función c_i^* es no nula cuando no se cumple y cero cuando sí.

La función de penalización consiste en una suma de la función objetivo original y un término de penalización, i. e., $P(x) = J(x) + h(c^*(x))$ tenemos que $h(c^*(x))$ será una función definida en $\mathbb{R}^m$ y satisface $h(0) = 0$, $\lim_{||c|| \to \infty} h(c) = \infty$. Un ejemplo será $P_2(x) = J(x) + \eta||c^*(x)||_2^2$ con $\eta > 0$ constante, parámetro de penalización. También tendremos $P_1(x) = J(x) + \eta||c^*(x)||_1$ y $P_\infty(x) = J(x) + \eta||c^*(x)||_\infty$.

Método de penalización simple

Este método permite obtener el mínimo del problema de optimización original (2.2) minimizando una serie de funciones de penalización. Consideremos la función de penalización simple $P_\eta(x) = J(x) + \eta||c^*(x)||^\alpha$, donde $\eta > 0$ parámetro de penalización destinado a tender a infinito (o hacerse grande), $\alpha > 0$ constante y $||.||$ una norma dada en $\mathbb{R}^m$. Escribiremos $x(\eta)$ solución de $\min_{x\in\mathbb{R}^n} P_\eta(x)$.

Lema 2.37. *Sea $0 < \sigma_1 < \sigma_2$. Entonces*

$$\begin{aligned} P_{\sigma_1}(x(\sigma_1)) \leq & P_{\sigma_2}(x(\sigma_2)), \\ J(x(\sigma_1)) \leq & J(x(\sigma_2)), \\ ||c^*(x(\sigma_1))|| \geq & ||c^*(x(\sigma_2))||. \end{aligned}$$

Demostración. Demostración en el capítulo 10, lema 10.2.1, página 461 de [4]. □

Lema 2.38. *Sea $\delta = ||c^*(x(\eta))||$. Entonces $x(\eta)$ es también solución del problema*

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} & J(x) \\ \textit{sujeto a} & ||c^*(x)|| \leq \delta. \end{cases}$$

Demostración. Demostración en el capítulo 10, lema 10.2.2, página 461 de [4]. □

Por la definición de $c^*(x)$, el problema original (2.2) se puede reescribir

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} & J(x) \\ \text{sujeto a} & ||c^*(x)|| = 0. \end{cases}$$

De manera que si δ es lo suficientemente pequeño obtenemos una aproximación y $x(\eta)$ se podrá considerar una aproximación de la solución del problema original (2.2). De hecho si $c^*(x(\eta)) = 0$, $x(\eta)$ será la solución del problema original.

La idea básica del método es que el parámetro de penalización η se incremente en cada iteración hasta que $||c^*(x(\eta))||$ sea menor que una tolerancia dada.

Algoritmo 2.39.

Paso 0 Damos $x_0 \in \mathbb{R}^n$, $\eta_1 > 0$, $\epsilon \geq 0$ y inicializamos $k := 1$.

Paso 1 Buscamos la solución $x(\eta_k)$ de $\min_{x\in\mathbb{R}^n} P_{\eta_k}(x)$ empezando en x_k.

Paso 2 Si $||c^*(x(\eta_k))|| \leq \epsilon$ terminamos.

Paso 3 Actualizamos $x_{k+1} = x(\eta_k)$, $\eta_{k+1} = 10\eta_k$; $k := k + 1$ y volvemos al Paso 1.

Observación 2.40. Podemos escoger η_k dependiendo de la dificultad de la minimización de la función de penalización en la iteración k-ésima ($\eta_{k+1} = C\eta_k$, $C > 1$).

Teorema 2.41. *Supongamos que la tolerancia, ϵ del algoritmo satisface $\epsilon > \min_{x\in\mathbb{R}^n} ||c^*(x)||$ entonces el algoritmo debe terminar de manera finita.*

Demostración. Véase el capítulo 10, teorema 10.2.4, página 463 de [4]. □

Teorema 2.42. *Si el algoritmo no termina de manera finita, entonces $\min_{x\in\mathbb{R}^n} ||c^*(x)|| \geq \epsilon$ y $\lim_{k\to\infty} ||c^*(x(\eta_k))|| = \min_{x\in\mathbb{R}^n} ||c^*(x)||$, y cualquier punto de acumulación $\hat{x}$ de $\{x(\eta_k)\}$ es solución del problema*

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} & J(x) \\ sujeto\ a & ||c^*(x)|| = \min\limits_{y\in\mathbb{R}^n} ||c^*(y)||. \end{cases}$$

Demostración. Véase el capítulo 10, teorema 10.2.5, página 463 de [4]. □

De los dos teoremas anteriores obtenemos

Corolario 2.43. *Si el problema tiene puntos factibles ($\min_{x\in\mathbb{R}^n} ||c^*(x)|| = 0$). Entonces o el algoritmo encuentra de manera finita la solución o cualquier punto de acumulación de la secuencia generada será solución del problema original (2.2).*

Capítulo 3

Control óptimo de sistemas discretos

En este capítulo estudiaremos el control óptimo de sistemas. Empezaremos dando la definición del problema de control para pasar a dar su planteamiento como problemas de optimización con y sin restricciones. Obtendremos las ecuaciones que deben satisfacer los controles óptimos y las relacionaremos con el sistema proporcionado por los multiplicadores de Lagrange en el capítulo anterior. Veremos algún método de resolución de problemas de control que será implementado y usado para resolver problemas académicos. Por último se aplicarán estas técnicas a problemas de interés tales como la resolución de un problema de control en EDO tras haber sido discretizado o el entrenamiento de una red neuronal de clasificación.

3.1. Planteamiento del problema

Con frecuencia nos enfrentamos a problemas de minimización

$$\min_{u \in S} \quad \tilde{J}(u)$$

donde la expresión del funcional $\tilde{J}$ es compleja pero que admite una reescritura

$$\tilde{J}(u) = J(y(u), u)$$

donde $y(u)$ es la solución de

$$F(y(u), u) = 0. \tag{3.1}$$

En estos casos $u \in \mathbb{R}^n$ recibe el nombre de *variable de control* mientras que $y \in \mathbb{R}^m$ es la llamada *variable de estado* definida por la *ecuación de estado* (3.1). Tendremos $F : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$ diferenciable, denotaremos por $y_u \equiv y(u)$ a la única (por hipótesis)

solución de la ecuación. Así el problema será

$$\begin{cases} \min\limits_{u\in\mathbb{R}^n} \quad \tilde{J}(u) := J(y_u, u) \\ \text{donde } y_u \text{ es la única solución de } F(y_u, u) = 0, \end{cases} \tag{3.2}$$

con $\tilde{J}$ también diferenciable.

Vectorialmente tenemos

$$y = (y_1, y_2, \dots y_m),$$
$$F(y, u) = (f_1(y, u), f_2(y, u), \dots, f_m(y, u))^T$$

identificaremos f_i como una función que dependerá de la variable de control y de la variable de estado.

3.1.1. Planteamiento como problema de optimización con restriciones

El problema (3.2) se puede ver como el siguiente problema de optimización con restricciones de igualdad,

$$\begin{cases} \min\limits_{(y,u)} \quad J(y, u) \\ \text{sujeto a} \quad F(y, u) = 0 \end{cases} \tag{3.3}$$

Con la nueva formulación (3.3) será condición necesaria que las parciales del Lagrangiano con respecto a todas la variables se anulen (Teorema 2.7). $\lambda \in \mathbb{R}^m$ será el vector que contiene a los multiplicadores de Lagrange, en el marco de la teoría de control óptimo recibe el nombre de estado adjunto, y el Lagrangiano será

$$\mathcal{L}(y, u, \lambda) = J(y, u) - \lambda^T F(y, u).$$

Así tendremos

$$\begin{cases} d_\lambda \mathcal{L}(y, u, \lambda) = -F(y, u) = 0 & \text{Ecuación de estado} \\ d_y \mathcal{L}(y, u, \lambda) = d_y J(y, u) - \lambda^T d_y F(y, u) = 0 & \text{Sistema adjunto} \\ d_u \mathcal{L}(y, u, \lambda) = d_u J(y, u) - \lambda^T d_u F(y, u) = 0 & \text{Condición de optimalidad} \end{cases} \tag{3.4}$$

La terminología usada aquí se entenderá más tarde.

Una manera más eficiente de ver esto es de manera vectorial y matricial, que nos ayudará sobre todo después a entender el método de resolución y a aplicarlo a casos más concretos.

La ecuación de estado la podremos ver como $F(y, u) = 0$ y teniendo en cuenta su formulación vectorial se tendrá que cumplir

$$f_i(y, u) = 0 \quad \forall i \in \{1, \dots, m\}. \tag{3.5}$$

El sistema adjunto lo podemos ver como $d_yF^T(y,u)\lambda = d_yJ^T(y,u)$. Obviamos la dependencia de y y de u para simplificar la notación y tenemos

$$d_yF(y,u) = \begin{pmatrix} \partial_{y_1}f_1 & \partial_{y_2}f_1 & \dots & \partial_{y_m}f_1 \\ \partial_{y_1}f_2 & \partial_{y_2}f_2 & \dots & \partial_{y_m}f_2 \\ \vdots & \vdots & \ddots & \vdots \\ \partial_{y_1}f_m & \partial_{y_2}f_m & \dots & \partial_{y_m}f_m \end{pmatrix}$$

$$d_yJ(y,u) = (\partial_{y_1}J, \partial_{y_2}J, \dots, \partial_{y_m}J)$$

Entonces podemos expresar $d_yF^T(y,u)\lambda = d_yJ^T(y,u)$ como

$$\begin{pmatrix} \partial_{y_1}f_1 & \partial_{y_2}f_1 & \dots & \partial_{y_m}f_1 \\ \partial_{y_1}f_2 & \partial_{y_2}f_2 & \dots & \partial_{y_m}f_2 \\ \vdots & \vdots & \ddots & \vdots \\ \partial_{y_1}f_m & \partial_{y_2}f_m & \dots & \partial_{y_m}f_m \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{pmatrix} = \begin{pmatrix} \partial_{y_1}J \\ \partial_{y_2}J \\ \vdots \\ \partial_{y_m}J \end{pmatrix} \tag{3.6}$$

En cuanto a la condición de optimalidad tenemos que la podemos expresar como $d_uJ^T(y,u) - d_uF^T(y,u)\lambda = 0$ vectorialmente y omitendo como antes la dependencia tenemos

$$d_uJ(y,u) = (\partial_{u_1}J, \partial_{u_2}J, \dots, \partial_{u_n}J)$$

$$d_uF(y,u) = \begin{pmatrix} \partial_{u_1}f_1 & \partial_{u_2}f_1 & \dots & \partial_{u_n}f_1 \\ \partial_{u_1}f_2 & \partial_{u_2}f_2 & \dots & \partial_{u_n}f_2 \\ \vdots & \vdots & \ddots & \vdots \\ \partial_{u_1}f_m & \partial_{u_2}f_m & \dots & \partial_{u_n}f_m \end{pmatrix}$$

Entonces tendremos

$$\begin{pmatrix} \partial_{u_1}J \\ \partial_{u_2}J \\ \vdots \\ \partial_{u_n}J \end{pmatrix} - \begin{pmatrix} \partial_{u_1}f_1 & \partial_{u_1}f_2 & \dots & \partial_{u_1}f_m \\ \partial_{u_2}f_1 & \partial_{u_2}f_2 & \dots & \partial_{u_2}f_m \\ \vdots & \vdots & \ddots & \vdots \\ \partial_{u_n}f_1 & \partial_{u_n}f_2 & \dots & \partial_{u_n}f_m \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{3.7}$$

Observación 3.1. Tomamos todas las fórmulas traspuestas con respecto a lo que aparece en (3.4) para que nos coincidan con como vamos a calcular el gradiente de $\tilde{J}$ en la sección 3.1.2 donde se reformulará el problema como un problemas de optimización sin restricciones.

3.1.2. Planteamiento como problema de optimización sin restricciones

También podemos ver el problema (3.2) como un problema de optimización sin restricciones. En este caso habría que obtener en primer lugar y_u resolviendo $F(y_u, u) = 0$ y tomando luego el problema

$$\min_u \tilde{J}(u) = J(y, u) \tag{3.8}$$

En este caso tenemos ahora que una condición necesaria será $\nabla \tilde{J} = 0$. A parte de por esto nos resultará de especial interés el cálculo del gradiente ya que para la mayoría de métodos de resolución para problemas de optimización sin restricciones vistos en el capitulo 1 es necesario. Veremos dos formas distintas de calcular $\nabla \tilde{J}$.

Cálculo del gradiente con el linelizado

Ésta se podría considerar la forma más directa de hacer el cálculo. Tenemos que $\nabla \tilde{J}(u) = d_u \tilde{J}^T(u)$ calcularemos entonces

$$d_u \tilde{J}(u) = d_y J(y_u, u) d_u y_u + d_u J(y_u, u) \tag{3.9}$$

en donde todo se obtendría de manera relativamente fácil a excepción de $d_u y_u$. Para calcular $d_u y_u$ se empieza derivando implícitamete la ecuación de estado (3.1) con lo cual obtenemos $d_y F(y_u, u) d_u y_u + d_u F(y_u, u) = 0$, que nos proporciona el sistema linealizado

$$d_y F(y_u, u) d_u y_u = -d_u F(y_u, u). \tag{3.10}$$

En este sistema podemos obtener $d_y F(y_u, u)$ y $d_u F(y_u, u)$, por tanto resolviendo la ecuación conseguiríamos $d_u y_u$. Tendríamos entonces todo lo necesario para el cálculo de (3.9) y por tanto también para obtener el gradiente.

En ocasiones puede resultar muy costosa la resolución de (3.10) en estas ocasiones resultará más adecuado calcular el gradiente como explicamos a continuación.

Cálculo del gradiente con el adjunto

Tenemos al igual que antes que $\nabla \tilde{J}(u) = d_u \tilde{J}^T(u)$ por tanto querremos de nuevo usar (3.9) para calcular $d_u \tilde{J}(x)$ pero en este caso abordaremos el problema con $d_u y_u$ de manera diferente. Buscaremos trasformar $d_y J(y_u, u) d_u y_u$ en algo que resulte más fácil calcular ($d_u J(y_u, u)$ ya aparece en la fórmula (3.7) y por tanto querremos substituir el otro sumando por $-\lambda^T d_u F(y_u, u)$ para que coincida).

Usamos en primer lugar el sistema linealizado 3.10. Multiplicamos escalarmente este sistema por $\lambda^T \in \mathbb{R}^m$ arbitrario y obtenemos

$$\lambda^T d_y F(y_u, u) d_u y_u = -\lambda^T d_u F(y_u, u).$$

La idea es considerar un λ tal que coincidan $d_y J(y_u, u) d_u y_u$ y $\lambda^T d_y F(y_u, u) d_u y_u$ para reemplazarlos por $-\lambda^T d_u F(y_u, u)$.

Tomamos λ definido por $d_y F^T(y_u, u)\lambda = d_y J^T(y_u, u)$ que coincide con el sistema adjunto y es un sistema lineal. En este caso, llamando $\lambda_u \equiv \lambda(u)$ a la solución del sistema obtenemos

$$d_u \tilde{J}(u) = -\lambda_u^T d_u F(y_u, u) + d_u J(y_u, u)$$

y por tanto

$$\nabla \tilde{J}(u) = d_u \tilde{J}^T(u) = -d_u F^T(y_u, u)\lambda_u + d_u J^T(y_u, u). \tag{3.11}$$

Tenemos entonces que podemos calcular el gradiente resolviendo la ecuación de estado (3.1) para calcular y, el sistema (3.6) para calcula λ y el gradiente con (3.11).

3.2. Método de resolución

Para la resolución del problema resultará interesante considerar el planteamiento como problema de optimización sin restricciones. Usaremos entonces los métodos vistos en el capítulo 1 para este tipo de problemas calculando el gradiente de alguna de las formas vistas en la sección anterior.

De forma general tendremos que calcular el gradiente con uno de los dos métodos anteriores para después pasar a usar uno de los métodos de resolución del capítulo 1. Además dentro de los métodos tendremos que calcular el paso en cada iteración para lo cual también tendremos que escoger una de las reglas vistas en la correspondiente sección.

Vamos entonces a crear un programa que nos permita solucionar el problema. Buscaremos que sea aplicable a casos académicos con los que comprobaremos su funcionamiento. Nos permitirá escoger como calcular el gradiente (linealizado o adjunto), que método usar (gradiente o gradiente conjugado) y como calcular el paso (Armijo, Wolfe o paso fijo). Debido a la longitud de este programa a continuación se incluye solo el código de uno de los casos que hemos hablado. El código completo se tendrá como material adicional. Por tanto vemos ahora un programa que resuelve el problema de control calculando el gradiente con el linealizado, usando el método del gradiente conjugado y la regla de Armijo.

Código control óptimo (linelizado, gradiente conjugado y Armijo)

```
1 function [u,y,j,conv,ng,d]=conopt_lin_gradconj_arm(n,m,J,duJ,dyJ, ...
```

```
2 F,duF,dyF,u0,kmax,tol,tau,rho,beta,itmax)
3 %[u,y,j,conv,ng,d]=conopt_lin_gradconj_arm(n,m,J,duJ,dyJ, ...
4 %F,duF,dyF,u0,kmax,tol,tau,rho,beta,itmax)
5 %Programa que nos resuleve el problema de control optimo usando el sistema
6 %linealizado para calcular el gradiente, el metodo del gradiente conjugado
7 %para optimizar y la regla de Armijo para el calculo de paso. Los
8 %argumentos de entrada seran:
9 %n              : Longitud del vector u.
10 %m             : Longitud del vector y.
11 %J             : Funcional objetivo dependiendo de y y u.
12 %duJ           : Vector fila con las derivadas de J con respecto a u.
13 %dyJ           : Vector fila con las derivadas de J con respecto a y.
14 %F             : Funcion vectorial con longitud m dependiendo de y y u.
15 %duF           : Matriz m*n con las derivadas de F con respecto a u.
16 %dyF           : Matriz m*m con las derivadas de F con respecto a y.
17 %u0            : Valor inicial de la variable de control.
18 %kmax          : Numero maximo de iteraciones.
19 %tol           : Tolerancia para el test de parada.
20 %tau, rho, beta : Parametros que definen los criterios de la regla de
21 %                 Armijo. Se supone que 0 < tau, 0 < rho <1/2 y
22 %                 0 < beta < 1.
23 %itmax         : Numero maximo de iteraciones en el calculo de paso.
24 %La funcion nos devolvera:
25 %u    : Valor de la variable de control en el minimo.
26 %y    : Valor de la variable de estado en el minimo.
27 %j0   : Valor del funcional objetivo en el minimo.
28 %conv : Nos indica si se ha llegado o no a la convergencia "1" si, "0" no.
29 %ng   : Valor de la norma del gradiente en la ultima iteracion, deberia
30 %       estar cerca de cero si se da la convergencia.
31 %d    : Valor de la ultima direccion de descenso.
32
33 %Inicializacion
34 u=u0;
35 conv=0;
36 x0=ones(1,m);
37 %Calculo de yu
```

```
38 Fy=@(y) F(y,u);
39 y=fsolve(Fy,x0);
40 % Calculo del gradiente con el linealizado
41 duyu=dyF(y,u)\-duF(y,u);
42 g=(dyJ(y,u)*duyu+duJ(y,u)).';
43 % Metodo del gradiente conjugado
44 d=-g.';
45 ng=norm(d);
46 if(ng<tol)
47 ____conv=1;
48 ____j=J(y,u);
49 ____return;
50 end
51 for K=1:kmax
52 ____% Calculo de paos, regla de Armijo.
53 ____index=0;
54 ____j0=J(y,u);
55 ____jp0=d*g;
56 ____rtau=j0+rho*jp0*tau;
57 ____utau=u+tau*d;
58 ____jtau=J(y,utau);
59 ____if (jtau<=rtau)&&(rtau<j0)
60 ____,____alpha=tau;
61 ____,____index=1;
62 ____else
63 ____,____for i0=1:itmax
64 ____,____,____bt=beta^i0*rho;
65 ____,____,____rbt=j0+rho*jp0*bt;
66 ____,____,____ubt=u+bt*d;
67 ____,____,____jbt=J(y,ubt);
68 ____,____,____if (rbt>=jbt)
69 ____,____,____,____alpha=bt;
70 ____,____,____,____index=1;
71 ____,____,____,____break
72 ____,____,____end
73 ____,____end
```

```
74        if index~=1
75            error('No se ha podido calcular el paso')
76        end
77    end
78    % Actualizacion
79    u=u+alpha*d;
80
81    % Calculo de yu
82    Fy=@(y) F(y,u);
83    y=fsolve(Fy,x0);
84    % Calculo del gradiente con el linelizado
85    duyu=dyF(y,u)\-duF(y,u);
86    g1=(dyJ(y,u)*duyu+duJ(y,u)).';
87    ng=norm(g1);
88    if(ng<tol)
89        conv=1;
90        j=J(y,u);
91        return;
92    end
93    beta=norm(g1)^2/norm(g)^2;
94    g=g1;
95    d=-g.'+beta*d;
96 end
97 error('No se alcanzo la convergencia')
```

Veamos ahora que el programa funciona con un ejemplo académico que podemos resolver a mano. Consideramos el problema de encontrar $(u_1, u_2)^T \in \mathbb{R}^2$ que minimice la función:

$$\tilde{J}(u) := J(y_u, u) = 2y_1y_2 + u_1^2 + u_2^2,$$

siendo $y_u = (y_1, y_2)^T \in \mathbb{R}^2$ la solución del problema no lineal:

$$\begin{cases} y_1^3 = -(1 + u_1^2 + u_2^2), \\ y_1y_2 = -u_2. \end{cases}$$

En este caso tenemos:

$$F(y,u) = \begin{pmatrix} y_1^3 + u_1^2 + u_2^2 + 1 \\ y_1y_2 + u_2 \end{pmatrix}.$$

Calculamos las matrices jacobianas parciales:

$$\partial_y F = \begin{pmatrix} 3y_1^2 & 0 \\ y_2 & y_1 \end{pmatrix}, \quad \partial_u F = \begin{pmatrix} 2u_1 & 2u_2 \\ 0 & 1 \end{pmatrix},$$

$$\partial_y J = (2y_2,\ 2y_1), \quad \partial_u J = (2u_1, 2u_2).$$

El sistema de optimalidad viene dado por:

$$\left.\begin{aligned} y_1^3 + u_1^2 + u_2^3 + 1 = 0 \\ y_1 y_2 + u_2 = 0 \end{aligned}\right| \quad \text{Sitema de estado } (S_E)$$

$$\left.\begin{pmatrix} 3y_1^2 & y_2 \\ 0 & y_1 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} 2y_2 \\ 2y_1 \end{pmatrix}\right| \quad \text{Sistema adjunto } (S_A)$$

$$\left.\begin{pmatrix} 2u_1 & 0 \\ 2u_1 & 1 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} 2u_1 \\ 2u_2 \end{pmatrix}\right| \quad \text{Condición de potimalidad } (C_O)$$

Buscamos ahora la solución de manera analítica. Si $u_1 \neq 0$ entonces

$$C_{O,1} \Rightarrow \lambda_1 = 1,$$

$$C_{O,2} \Rightarrow \lambda_2 = 0.$$

Teniendo en cuenta esto en S_A:

$$y_1 = 0;\ 3y_1^2 = 2y_2 \Rightarrow y_2 = 0.$$

Usándolo en S_E

$$u_1^2 + u_2^2 = -1,$$

lo cual no es posible para $u \in \mathbb{R}^2$. Tenemos entonces que $u_1 = 0$ por tanto podemos reescribir S_E como

$$y_1^3 + u_2^2 + 1 = 0,$$

$$y_1 y_2 + u_2 = 0.$$

De igual modo tenemos que S_A será

$$3y_1^2\lambda_1 + y_2\lambda_2 = 2y_2,$$

$$y_1\lambda_2 = 2y_1.$$

Finalmente de C_O nos quedará

$$2u_2\lambda_1 + \lambda_2 = 2u_2.$$

Si $y_1 \neq 0 \Rightarrow \lambda_2 = 2$, usando la segunda ecuación de S_A. Además por C_O ahora tendremos $u_2(1-\lambda_1) = 1$ y por la primera ecuación de S_A tenemos $3y_1^2\lambda_1 = 0 \Rightarrow \lambda_1 = 0 \Rightarrow u_2 = 1$. Usando esto en S_E

$$y_1^3 = -2 \Rightarrow y_1 = -\sqrt[3]{2} \Rightarrow y_2 = -\frac{1}{y_1} = \frac{1}{\sqrt[3]{2}}.$$

Tenemos entonces la solución

$$y = (-\sqrt[3]{2}, \frac{1}{\sqrt[3]{2}})^T$$
$$u = (0,1)^T$$
$$z = (0,2)^T$$

Si $y_1 = 0 \Rightarrow u_2 = 0$ y $u_2^2 + 1 = 0$ lo cual no es posible y hemos obtenido una única solución el problema de optimalidad.

Veamos la resolución con el código. Para resolver este problema académico tenemos que introducir los siguientes comandos

```
n=2;
m=2;
J=@(y,u) 2*y(1)*y(2)+u(1)^2+u(2)^2;
duJ=@(y,u) [2*u(1),2*u(2)];
dyJ=@(y,u) [2*y(2),2*y(1)];
F=@(y,u) [y(1)^3+1+u(1)^2+u(2)^2;y(1)*y(2)+u(2)];
duF=@(y,u) [2*u(1),2*u(2);0,1];
dyF=@(y,u)[3*y(1)^2,0;y(2),y(1)];
u0=[2,2];
kmax=100;
tol=1e-6;
tau=0.25;
rho=0.25;
beta=0.25;
itmax=100;


[u,y,j,conv,ng,d]=conopt_lin_gradconj_arm(n,m,J,duJ,dyJ,F,duF,dyF, ...
u0,kmax,tol,tau,rho,beta,itmax);
```

lo cual nos da como resultado

$u = (0, 1)$ Valor de la variable de control en el mínimo.

$y = (-1'2599, 0'7937)$ Valor de la variable de estado en el mínimo.

$j = -1$ Valor del funcional objetivo en el mínimo.

$conv = 1$ Nos indica que se ha alcanzado la convergencia.

$ng = 7'9643e - 07$ Valor de la norma del gradiente en la última iteración.

$d = (-0'1007e - 05, -0'0504e - 05)$ Valor del descenso en la última iteración.

3.3. Aplicaciones

En está ultima sección del capítulo veremos algunas aplicaciones interesantes del control óptimo. Tendremos en primer lugar una descripción de los denominados sistemas dinámicos discretos. Pasaremos a continuación a ver como se les aplicaría el control óptimo y como esta aplicación nos dará un método de resolución propio para estos caso. Finalmente daremos dos casos en los que se aplicará el control óptimo relacionados con los estos sistemas.

3.3.1. Sistemas dinámicos discretos

En el caso de los sistemas dinámicos discretos tendremos un problema como el descrito en (3.2) pero tendremos ciertas condiciones especiales para la ecuación de estado (3.1). Al tratar ahora con un sistema dinámico discreto tendremos que a medida que vayamos avanzando por las variables estas estarán definidas en función de las anteriores. Recordemos que vectorialmente teníamos

$$y = (y_1, y_2, \dots y_m),$$

$$F(y, u) = (f_1(y, u), f_2(y, u), \dots, f_m(y, u))^T$$

identificando f_i como una función que dependerá de la variable de control y de la variable de estado. En este caso identificaremos f_i como la función que definirá la variable y_i y que dependerá de la variable de control y a lo sumo de las variables y_j con $j \leq i$. Esto se justifica por ser un sistema dinámico discreto un sistema que evoluciona a través de unos instantes. En el instante i la variable de estado valdrá y_i. Además no tendremos información sobre los instantes siguientes.

Método de resolución

Veremos ahora un estudio de como se obtendrá la solución del problema en este caso particular por el método del gradiente calculándose este con la fórmula obtenida usando el

adjunto (3.7). La dependencia de las funciones f_i nos dará una estructura muy particular de la matriz que interviene en (3.6) y de las funciones de (3.5).

Ahora bien, lo que vamos a usar es el método del gradiente y la variable en la cual vamos a buscar el mínimo sera la variable de control u. Por tanto lo primero que tendremos que hacer será dar u^0 inicial, el cual usaremos para construir la sucesión que convergerá al mínimo. No usaremos u_k para nombrar a los elemento de la sucesión pues esta es la notación que estamos usando para lo elementos del vector u. En su lugar usaremos u^k para indicar que estamos hablando del valor de u en concreto en la iteración k y actualizaremos sobre u entendiendo que esto creará una sucesión convergente a la solución. Aclarado esto podemos pasar a la parte más especifica del método.

En cada iteración k, por como tenemos definida la función $F(y,u)$, sabemos que f_1 solo dependerá de y_1 y u. Al tener dado el valor de u como u^k tendremos que $F(y,u^k)=0$ en su primera coordenada, $f_1(y_1,u^k)=0$, nos dará el valor de y_1, y_1^k. Esto tiene sentido puesto que en el instante inicial tendremos que tener el valor de la variable de estado. No lo podremos dar en función de los anteriores pues no hay ninguno. A partir de esto, por como tenemos definidos los f_i podemos obtener de manera recurrente los y_i^k pues tenemos el valor de la variable de control y los valores de la variable de estado el los instantes anteriores. Obtendremos así todos los valores de la variable de estado y^k.

A continuación tendremos que calcular λ^k y lo calcularemos como la solución de (3.6). Ahora bien teniendo en cuenta de nuevo como asumimos que estaban construidas las funciones f_i tendremos $\partial_{y_t} f_l = 0$ para $t > l$. Esto nos dará una construcción de la ecuación (3.6) que nos permitirá calcular los λ_i de manera recurrente retrógrada. Tendremos la representación matricial

$$\begin{pmatrix} \partial_{y_1} f_1 & \partial_{y_1} f_2 & \dots & \partial_{y_1} f_m \\ 0 & \partial_{y_2} f_2 & \dots & \partial_{y_2} f_m \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \partial_{y_m} f_m \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{pmatrix} = \begin{pmatrix} \partial_{y_1} J \\ \partial_{y_2} J \\ \vdots \\ \partial_{y_m} J \end{pmatrix}$$

de aquí obtendremos las ecuaciones

$$\begin{aligned} \partial_{y_m} f_m \lambda_m &= \partial_{y_m} J \\ &\vdots \\ \partial_{y_2} f_2 \lambda_2 + \dots + \partial_{y_2} f_m \lambda_m &= \partial_{y_2} J \\ \partial_{y_1} f_1 \lambda_1 + \partial_{y_1} f_2 \lambda_2 + \dots + \partial_{y_1} f_m \lambda_m &= \partial_{y_1} J \end{aligned} \tag{3.12}$$

en las cuales se observa de manera clara la recurrencia retrógrada de la que habíamos hablado. Teniendo en cuenta que $\partial_{y_t} f_l$ y $\partial_{y_t} J$ estarán evaluadas en u^k y y^k podremos

calcular λ^k y tendremos todo lo necesario para calcular el gradiente $\bigtriangledown \tilde{J}(u^k)$ mediante la fórmula (3.7).

Una vez hecho esto podremos comprobar si se cumple el criterio de convergencia $||\bigtriangledown \tilde{J}(u^k)|| < \epsilon$ para un cierto $\epsilon > 0$, tolerancia dada. Si el criterio de convergencia no se cumple podremos usar alguno de los métodos vistos en el primer capítulo para calcular el paso, α^k y actualizaremos $u^{k+1} = u^k - \alpha \bigtriangledown \tilde{J}(u^k)$; $k = k+1$. Después de esto volveremos a empezar el proceso calculando los y_i^k.

Veremos ahora el algoritmo de manera más esquematizada para que nos quede más claro.

Algoritmo 3.2.

Paso 0 *Damos u^0, $\epsilon > 0$ la tolerancia e inicializamos $k := 0$.*

Paso 1 *Calculamos los y_i^k de manera recurrente con las funciones $f_i(y_i, ..., y_1, u^k)$ igualadas a cero.*

Paso 2 *Calculamos λ^k de manera recurrente retrograda con las ecuaciones 3.12.*

Paso 3 *Calculamos $\bigtriangledown \tilde{J}(u^k)$ con la fórmula 3.7.*

Paso 4 *Si se cumple $||\bigtriangledown \tilde{J}(u^k)|| < \epsilon$ terminamos y u^k es el mínimo.*

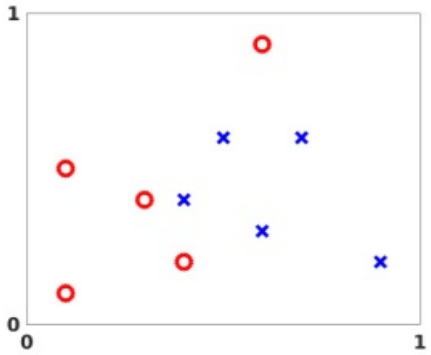
Paso 5 *Calculamos el paso, α^k, de alguna de las formas vistas en el capítulo 1.*

Paso 6 *Actualizamos $u^{k+1} = u^k - \alpha^k \bigtriangledown \tilde{J}(u^k)$, $k := k+1$ y volvemos al Paso 1.*

3.3.2. Aprendizaje de una red neuronal artificial

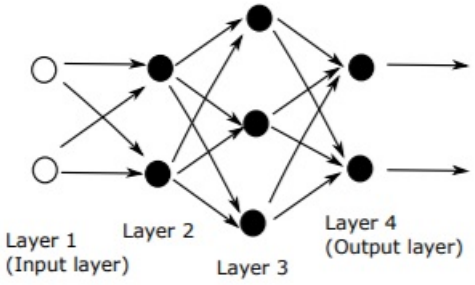
Vamos a dar ahora un ejemplo de uso del control óptimo de sistemas en el aprendizaje de redes neuronales artificiales. Este ejemplo está sacado de el artículo [9]. Buscamos en primer lugar dar una descripción del problema general del aprendizaje de una red neuronal artificial relacionándolo con lo visto anteriormente. Finamente damos un código basado en el que aparece en el artículo pero generalizado.

Consideramos un conjunto de puntos divididos en dos categorías, A y B. Lo que queremos es construir una función que dado un punto cualquiera nos lo clasifique dentro de una de estas categorías. Basaremos la red en la función $\sigma(x) = \frac{1}{1+e^{-x}}$. Podemos ver la función $\sigma(x)$ como una versión regular de una función salto, que replica el comportamiento de una neurona en el cerebro, activa (1) si la entrada es suficiente e inactiva (0) en otro caso. Además es conveniente que su derivada sea de la forma $\sigma'(x) = \sigma(x)(1-\sigma(x))$. El escalón y su posición en la función sigma se puede alterar escalando y cambiando el argumento o, en lenguaje de redes neuronales añadiendo un peso y desplazamiento a la entrada. Para



mantener una notación sencilla se define $\sigma : \mathbb{R}^m \to \mathbb{R}^m$ en $z \in \mathbb{R}^m$ de manera vectorial siendo $\sigma_i(z) = \sigma(z_i)$.

Tenemos ahora una serie de capas de neuronas. En cada capa, cada neurona devuelve un único numero real, que se pasa a cada neurona de la capa siguiente. Ahora cada neurona en esta capa crea su propia combinación de estos valores (operación afín) con un peso y desplazamiento propios y aplica la función sigma. Matemáticamente tenemos que si a es el vector de los números reales producidos por una capa el vector de salida de la siguiente tendrá la forma $\sigma(Wa + b)$. W será una matriz que contiene los pesos y b un vector con los desplazamientos. Sean n_i el número de neuronas de una capa y n_{i+1} el de la capa siguiente. Tendremos $a \in \mathbb{R}^{n_i}$ pues recoge los valores en las neuronas de la primera capa. $W \in \mathbb{R}^{n_{i+1} \times n_i}$ pues multiplicará a a y la dimensión del vector resultado de la multiplicación deberá ser el número de neuronas de la segunda capa. Además la dimensión de b también debe coincidir con el número de neuronas de la segunda capa, por tanto $b \in \mathbb{R}^{n+1}$.



Aplicando esto ahora a un caso en el que tenemos L capas hay que distinguir dos capas que se comportarán de manera diferente. La primera capa, la de entrada no tendrá una capa que le pase sus valores, contendrá los datos de entrada y por tanto el número de neuronas de esta capa coincidirá con la dimensión de estos. En la capa de salida, la L-

ésima no hay capa siguiente y estas neuronas proporcionan la salida final. Suponemos que la capa l, para $l = 1, 2, 3, \ldots, L$, contiene n_l neuronas. Usamos $W^l \in \mathbb{R}^{n_l \times n_{l-1}}$ para denotar la matriz de pesos en la capa l. Específicamente W^l_{jk} es el peso que la neurona j en la capa l aplica a la salida de la neurona k en la capa $l-1$. Analogamente, $b^l \in \mathbb{R}^n_l$ es el vector de desplazamientos para la capa l, así la neurona j de la capa l usa el desplazamiento b^l_j.

Dada una entrada $x \in \mathbb{R}^{n_1}$, podemos resumir la acción de la red siendo a^l_j la salida, o activación, de la neurona j en la capa l. Así tenemos

$$a^1 = x \in \mathbb{R}^{n_1} \tag{3.13}$$

$$a^l = \sigma(W^l a^{l-1} + b^l) \in \mathbb{R}^{n_l} \text{ para } l = 2, 3, \ldots, L. \tag{3.14}$$

Estás ecuaciones serán la relacionadas con el sistema de estado y son las que nos permitirán calcular las variables de estado de manera recurrente. Queda claro que con ellas y un algoritmo para llevar la entrada a través de la red obtendremos la salida $a^L \in \mathbb{R}^{n_L}$. Lo que nos interesará será tener unos pesos y desplazamientos que hagan que las salidas nos clasifiquen correctamente los puntos que introducimos, es decir, tener salida $(1, 0)$ si el punto está en la categoría A y $(0, 1)$ si está en la B.

Ahora suponemos que tenemos N datos, o puntos dato, en $\mathbb{R}^{n_L}$, $\{x^i\}_{i=1}^N$, para los que nos dan objetivos de salida $\{y(x^i)\}_{i=1}^N$ en $\mathbb{R}^{n_L}$, donde

$$y(x^i) = \begin{cases} \begin{bmatrix} 1 \\ 0 \end{bmatrix} & \text{si } x^i \text{ está en la categoría } A \\ \\ \begin{bmatrix} 0 \\ 1 \end{bmatrix} & \text{si } x^i \text{ está en la categoría } B. \end{cases}$$

Nuestra función de coste toma entonces la forma $Cost = \frac{1}{N} \sum_{i=1}^N \frac{1}{2} ||y(x^i) - a^L(x^i)||_2^2$ en donde para no alargarnos con la notación no se especifica que el coste es función de los pesos y desplazamientos. Aquí se introduce el factor $\frac{1}{2}$ por conveniencia.

Así tendríamos un problema de control óptimo. Las variables de control serán los pesos y desplazamientos, la variable de estado la salida a^L, el funcional objetivo será la función coste y a_L la solución del sistema compuesto por (3.13) y (3.14), que nos dará el sistema de estado. Para optimizar querremos usar $\bigtriangledown Cost$ pero esto puede suponer un gran coste computacional al contener un sumatorio a lo largo de los puntos dato. Usamos entonces el gradiente estocástico y en cada iteración escogeremos un punto x^i aleatoriamente y substituiremos la función $Cost$ por uno de sus sumandos $C_{x^i} = \frac{1}{2} ||y(x^i) - a^L(x^i)||_2^2$ para simplificar tomaremos $C = \frac{1}{2} ||y - a^L||_2^2$ evitando la dependencia de x^i.

Vamos a calcular a continuación derivadas parciales con respecto a la variables de control, pero para simplificar la notación es interesante introducir dos nuevos conjuntos

de variables. $z^l = W^l a^{l-1} + b^l \in \mathbb{R}^{n_l}$ para $l = 2, 3, \ldots, L$, donde z_j^k es la entrada escalada de la neurona j en la capa l. Así, $a^l = \sigma(z^l)$ con $l = 2, 3, \ldots, L$ y $\delta^l \in \mathbb{R}^{n_l}$ definido como $\delta_j^l = \frac{\partial C}{\partial x_j^l},\ 1 \leq j \leq n_l,\ 2 \leq l \leq L$. Esta expresión que se acostumbra a llamar error de la j-ésima neurona de la capa l es una cantidad intermedia que es útil en el análisis y en la computación. El termino error en este caso no es el más acertado pero cobra sentido porque la función coste solo purde minimizarse si todos los δ_j^l son cero (podremos observar que las variables δ son el estado adjunto en la terminología de control óptimo). Este paso se correspondería al del cálculo del estado adjunto usando el sistema adjunto, debido a las particularidades de este ejemplo no se corresponde exactamente con lo visto en el caso teórico.

Veamos entonces como calcular estos δ^l, que se corresponderán con los multiplicadores de Lagrange. Tenemos que $a^L = \sigma(z^L)$ entonces

$$\frac{\partial a_j^L}{\partial z_j^L} = \sigma'(z_j^L)$$

y por la definición de C,

$$\frac{\partial C}{\partial a_j^L} = \frac{\partial}{\partial a_j^L} \frac{1}{2} \sum_{k=1}^{n_L} (y_k - a_k^L)^2 = -(y_j - a_j^L).$$

Usando la regla de la cadena tenemos

$$\delta_j^L = \frac{\partial C}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L} = (a_j^L - y_j)\sigma'(z_j^L)$$

. Usamos de nuevo la regla de la cadena, esta vez para pasar de la derivada con respecto a z_j^j a las parciales con respecto a $\{z_k^{l+1}\}_k = 1^{n_{l+1}}$.

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_{k=1}^{n_{l+1}} \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_{k=1}^{n_{l+1}} \delta_j^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l}.$$

Ahora como $z^l = W^l a^{l-1} + b^l$ sabemos que z_k^{l+1} y z_j^l están conectados mediante $z_k^{l+1} = \sum_{s=1}^{n_l} W_{ks}^{l+1} \sigma(z_s^l) + b_k^{l+1}$. Por tanto

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = W_{kj}^{l+1} \sigma'(z_j^l)$$

y tenemos

$$\delta_j^l = \sum_{k=1}^{n_{l+1}} \delta_j^{l+1} W_{kj}^{l+1} \sigma'(z_j^l).$$

La fórmula anterior la podemos reajustar como

$$\delta_j^l = \sigma'(z_j^l) \left((W^{l+1})^T \delta^{l+1} \right).$$

Podemos ver entonces el sistema adjunto como

$$\delta_j^L = (a_j^L - y_j)\sigma'(z_j^L), \tag{3.15}$$

$$\delta_j^l = \sigma'(z_j^l)\left((W^{l+1})^T\delta^{l+1}\right). \tag{3.16}$$

y observamos como efectivamente nos permite calcular los δ^l de manera recurrente retrograda, como hacíamos con los multiplicadores de Lagrange.

Una vez calculados los δ_j^l pasamos al cálculo de las derivadas parciales de la función coste con respecto a las componentes de las variables de control. Con estas derivadas construiremos los gradientes con respecto a cada una de las variables de control que después usaremos en el método del gradiente estocástico con paso fijo. Sabemos que z_j^l está conectado a b_j^l por $z_j^l = (W^l\sigma(z^{l-1}))_j + b_j^l$. Tenemos que $\frac{\partial z_j^l}{\partial b_j^l} = 1$. Entonces usando la regla de la cadena

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{b_j^l} = \frac{\partial C}{\partial z_j^l} = \delta_j^l.$$

Finalmente tenemos $z_j^l = \sum_{k=1}^{n_{l-1}} W_{jk}^l a_k^{l-1} + b_j^l$, que nos da $\frac{\partial z_j^l}{\partial W_{jk}^l} = a_x^{l-1}$, independientemente de j y $\frac{\partial z_s^l}{\partial W_{jk}^l} = 0$ para $s \neq j$. Entonces aplicando la regla de la cadena de nuevo tenemos

$$\frac{\partial C}{\partial W_{jk}^l} = \sum_{s=1}^{n_l}\frac{\partial C}{\partial z_s^l}\frac{\partial z_s^l}{\partial W_{jk}^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial W_{jk}^l} = \delta_j^l a_k^{l-1}.$$

Resumiendo tenemos:

$$\begin{aligned}
&\delta_j^L = (a_j^L - y_j)\sigma'(z_j^L),\\
&\delta_j^l = \sigma'(z_j^l)\left((W^{l+1})^T\delta^{l+1}\right)_j,\\
&\frac{\partial C}{\partial b_j^l} = \delta_j^l,\\
&\frac{\partial C}{\partial W_{jk}^l} = \delta_j^l a_k^{l-1}.
\end{aligned}$$

Introduciendo ahora la matriz $D^l \in \mathbb{R}^{n_l \times n_l}$, matriz diagonal con $\sigma'(z_i^l)$ en la entrada (i,i) podemos reescribir

$$\begin{aligned}
&\delta^L = D^L(a^L - y),\\
&\delta^l = D^l\left(W^{l+1}\right)^T\delta^{l+1}.
\end{aligned}$$

Es interesante recordar además que cuando definimos la función σ vimos que su derivada era fácilmente calculable.

Con lo visto estamos ya en posición de dar un pseudocódigo usando el gradiente estocástico (Observación 1.39) con paso fijo η.

For counter $= 1$ a N_it

k escogido aleatoriamente en $\{1, \ldots, N\}$

x^k dato de entrenamiento

$a^1 = x^k$

For $l = 2 : L$

$z^l = W^l a^{l-1} + b^l$

$a^l = \sigma(z^l)$

$D^l = diag(\sigma'(z^l))$

end

For $l = L - 1 : 2$

$\delta^l = D^l (W^{l+1})^T \delta^{l+1}$

end

For $l = L : 2$

$W^l \to W^l - \eta \delta^l (a^{l-1})^T$

$b^l \to b^l - \eta \delta^l$

end

end

A continuación tenemos el código en Matlab, que como dijimos antes es una generalización del que se da el el artículo citado.

Aprendizaje de una red neuronal artificial

```
function [W,b]=rna_adjunto(nxc,clas,x1,x2,N_it,eta)
% rna_adjunto(nxc,clas,x1,x2,N_iter,eta)
% Usa el metodo del adjunto para entrenar una red neuronal artificial
%
% Vamos a introducir el numero de capas y el numero de neuronas en cada
% capa para que sea variable en el programa, las coordenadas de los
% elementos distinguiendo a que categoria pertenecen, el numero maximo de
% iteraciones que queremos que ejecute el programa y la tasa de
% aprendizaje.
```

```
%
% nxc:  Vector de longitud el numero de capas de nuestra RNA y que
%       contiene en cada elemento i el numero de neuronas de la capa
%       i.
% clas: Vector de longitud el numero distinto de clases a las que pueden
%       pertenecer los puntos dato. Cada elemento nos da el numero de
%       puntos dato en cada clase.
% x1:   Vector con las primeras componentes de los puntos dato. Su
%       longitud debe coincidir con la suma de los elemntos en clas, los
%       puntos deben estar ordenados por clase y las clases en el mismo
%       orden que el el vector clas. De manera que los clas(1) primeros
%       elementos de x1 correspondan con los puntos dato de la clase "1"
%       los clas(2) siguentes con los de la "2" y asi sucesivamente.
% x2:   Vector con las segundas coordenadas de los puntos dato. Debe estar
%       construido igual que x1 de forma que elemento i de x2 sea la
%       segunda coordenada del punto al que corresponde la primera
%       coordenada almacenada en x1(i).
% N_it: Numero de iteraciones que queremos llevar a cabo.
% eta:  Tasa de aprendizaje (tamanho de paso).


%Matriz de clasificacion de los elementos
y = zeros(length(clas),length(x1));
u=0;

for j=1:length(clas)
    l=u+1;
    u=u+clas(j);
    y(j,l:u)=1;
end

% Inicializacion de los pesos y desplazamientos
rng(5000);
capas=length(nxc);

w=cell(capas - 1,1);
```

```
for j=1:capas-1
    w(j)={0.5*randn(nxc(j+1),nxc(j))};
end
b=cell(capas-1,1);
for j=1:capas-1
    b(j)={0.5*randn(nxc(j+1),1)};
end

%Propagacion Forward and Backward
savecost = zeros(N_it,1); %Valor de la funcion coste en cada iteracion
N=length(x1);
for counter = 1:N_it
    k = randi(N); %Escoger un punto aleatorio para la evaluacion
    x = [x1(k); x2(k)];
    %Paso Forward
    a=cell(capas-1,1);
    x_act=x;
    for j=1:capas-1
        a(j)={activacion(x_act,cell2mat(w(j)),cell2mat(b(j)))};
        x_act=cell2mat(a(j));
    end

    %Paso Backward
    delta=cell(capas-1,1);
    delta(capas-1)={cell2mat(a(capas-1)).*(1-cell2mat(a(capas-1))) ...
    .*(cell2mat(a(capas-1))-y(:,k))};
    for j=capas-2:-1:1
        delta(j) = {cell2mat(a(j)).*(1-cell2mat(a(j))) ...
        .*(cell2mat(w(j+1))'*cell2mat(delta(j+1)))};
    end

    %Actualizacon metodo del gradiente
    w(1) = {cell2mat(w(1)) - eta*cell2mat(delta(1))*x'};
    b(1) = {cell2mat(b(1)) - eta*cell2mat(delta(1))};
    for j=2:capas-1
        w(j) = {cell2mat(w(j)) - eta*cell2mat(delta(j))*cell2mat(a(j-1))'};
```

```
82        b(j) = {cell2mat(b(j)) - eta*cell2mat(delta(j))};
83    end
84
85    %Visualizacion y gardado del progreso
86    newcost = cost(w,b)
87    savecost(counter) = newcost;
88 end
89 %Muestra la progresion del vector coste
90 save costvec
91 semilogy(1:1e4:N_it,savecost(1:1e4:N_it))
92 function costval = cost(w,b)
93    costvec = zeros(10,1);
94    for i = 1:10
95        x =[x1(i);x2(i)];
96        x_act=x;
97        for jk=1:capas-1
98            a(jk)={activacion(x_act,cell2mat(w(jk)),cell2mat(b(jk)))};
99            x_act=cell2mat(a(jk));
100       end
101       costvec(i) = norm(y(:,i) - cell2mat(a(capas-1)),2);
102   end
103   costval = norm(costvec,2)^2;
104 end
105 end
```

En está función se usa además la función de activación que será.

Función de activación

```
1 function y = activacion(x,W,b)
2 %y=activacion(x,W,b)
3 %Evalua la funcion sigma.
4 %
5 %x es el vector de entrada, y es el vector de salida
6 %W contiene los pesos, b contiene los shifts
7 %
8 %La componente i de y es activacion((Wx+b)_i)
9 %donde activacion(z) = 1/(1+exp(-z))
```

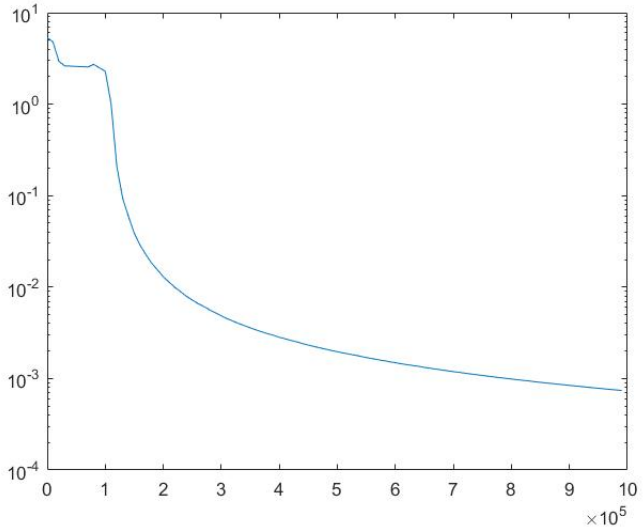
```
y = 1./(1+exp(-(W*x+b)));
```

Veremos ahora el funcionamiento de este programa con el ejemplo que se da en el articulo [9].

Prueba del programa

```
nxc=[2,2,3,2];
clas=[5,5];
x1 = [0.1,0.3,0.1,0.6,0.4,0.6,0.5,0.9,0.4,0.7];
x2 = [0.1,0.4,0.5,0.9,0.2,0.3,0.6,0.2,0.4,0.6];
N_it=1e6;
eta = 0.05;
rna_adjunto(nxc,clas,x1,x2,N_it,eta);
```

Está ejecución nos dará la siguiente gráfica que nos muestra en el eje x el número de iteraciones y en el eje y los valores de la función coste.



Observamos entonces como conseguimos reducir bastante la función coste aunque nos lleve un gran número de iteraciones. Esto nos indica que la red neuronal artificial habrá conseguido aprender a clasificar los puntos, pero este aprendizaje se podría decir que ha sido lento.

3.3.3. Discretización de un sistema con EDO

Estudiamos ahora un caso particular que nos permitirá relacionar lo estudiado con un problema de control en sistemas con EDO. Tenemos $y_0 \in \mathbb{R}$ dado. Denotamos $y =$

$(y_1, \ldots, y_n) \in \mathbb{R}^n$ y $u = (u_0, \ldots, u_{n-1}) \in \mathbb{R}^n$. Consideramos:

$$\begin{aligned} J : \mathbb{R}^n \times \mathbb{R}^n &\to \mathbb{R} \\ (y, u) &\rightsquigarrow J(y, u) = g_n(y_n) + \sum_{i=0}^{n-1} g_i(y_i, u_i), \end{aligned} \tag{3.17}$$

siendo

$$\begin{aligned} g_n : \mathbb{R} &\to \mathbb{R} \\ z &\rightsquigarrow g_n(z) \\ g_i : \mathbb{R} \times \mathbb{R} &\to \mathbb{R} \\ (z, v) &\rightsquigarrow g_i(z, v) \end{aligned}$$

funciones diferenciables escalares.

Tenemos el problema:

$$\begin{cases} min & J(y, u) \\ s.a: & y_{i+1} = h_i(y_i, u_i), \quad i \in \{0, 1, \ldots, n-1\} \end{cases}$$

donde

$$\begin{aligned} h_i : \mathbb{R} \times \mathbb{R} &\to \mathbb{R} \\ (z, v) &\rightsquigarrow h_i(z, v) \end{aligned}$$

son funciones diferenciables.

Tomamos ahora $f_i(y, u) = y_i - h_i(y_{i-1}, u_{i-1})$ en relación con lo visto en la sección anterior. El Lagrangiano será

$$\begin{aligned} \mathcal{L}(y, u, \lambda) =& J(y, u) - \lambda^T F(y, u) \\ =& g_n(y_n) + \sum_{i=0}^{n-1} g_i(y_i, u_i) - \sum_{i=1}^{n} \lambda_i (y_i - f_{i-1}(y_{i-1}, u_{i-1})). \end{aligned}$$

Tenemos entonces que la ecuación de estado será equivalente a

$$y_{i+1} = h_i(y_i, u_i), \quad i \in \{0, \ldots, n-1\}.$$

En cuanto al sistema adjunto deberemos calcular $\partial_{y_i} f_j(y, u))$ y $\partial_{y_i} J(y, u)$. Por la estructura de las funciones tenemos que

$$\partial_{y_i}(f_j(y, u)) = \begin{cases} 1 & \text{para } i = j, \\ -\partial_z h_i(y_i, u_i) & \text{para } i = j + 1, \\ 0 & \text{en otro caso.} \end{cases}$$

Por como definimos $J(y,u)$ tenemos

$$\partial y_i J(y,u) = \begin{cases} \partial_z g_i(y_i,u_i) & \text{para } i \in \{1,\dots,n-1\}, \\ g_n'(y_n) & \text{para } i = n. \end{cases}$$

Así obtenemos que el sistema adjunto será

$$\begin{pmatrix} 1 & -\partial_z h_1 & 0 & 0 & \dots & 0 \\ 0 & 1 & -\partial_z h_2 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & -\partial_z h_{n-2} & 0 \\ 0 & \dots & 0 & 0 & 1 & -\partial_z h_{n-1} \\ 0 & \dots & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{n-2} \\ \lambda_{n-1} \\ \lambda_n \end{pmatrix} = \begin{pmatrix} \partial_z g_1 \\ \partial_z g_2 \\ \vdots \\ \partial_z g_{n-2} \\ \partial_z g_{n-1} \\ g_n' \end{pmatrix}$$

En cuanto a la condición de optimalidad tenemos que calcular $\partial_{u_i} f_j(y,u)$ y $\partial_{u_i} J(y,u)$. Teniendo en cuenta que hemos definido $u = (u_0,\dots,u_{n-1})$ y de nuevo la estructura de la funciones calculamos

$$\partial_{u_i} f_j(y,u) = \begin{cases} -\partial_v h_i(y_i,u_i) & \text{para } i = j-1, \\ 0 & \text{en otro caso.} \end{cases}$$

Expresamos entonces la condición de optimalidad matricialmente como

$$\partial y_i J(y,u) = \begin{cases} \partial_z g_i(y_i,u_i) & \text{para } i \in \{1,\dots,n-1\}, \\ g_n'(y_n) & \text{para } i = n. \end{cases}$$

Así obtenemos que el sistema adjunto será

$$\begin{pmatrix} -\partial_v h_0 & 0 & 0 & \dots & 0 \\ 0 & -\partial_v h_1 & 0 & \dots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -\partial_v h_{n-2} & 0 \\ 0 & \dots & 0 & 0 & -\partial_v h_{n-1} \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{n-1} \\ \lambda_n \end{pmatrix} = \begin{pmatrix} \partial_v g_0 \\ \partial_v g_1 \\ \vdots \\ \partial_v g_{n-2} \\ \partial_v g_{n-1} \end{pmatrix}$$

$$\partial u_i J(y,u) = \partial_v h_i(y_i,u_i).$$

De estas representaciones podemos pasar a la siguiente forma de escritura:

Sistema de estado

$$\begin{aligned} &y_0 \text{ dado} \\ &y_{i+1} = h_i(y_i, u_i), \quad i \in \{0, 1, \ldots, n-1\} \end{aligned}$$

Sistema adjunto

$$\begin{aligned} &\lambda_n = g_n'(y_n) \\ &\lambda_i = \partial_z h_i(y_i, u_i)\lambda_{i+1} + \partial_z g_i(y_i, u_i), \quad i \in \{n-1, \ldots, 1\} \end{aligned}$$

Condición de optimalidad

$$-\partial_v h_{i-1}(y_{i-1}, u_{i-1})\lambda_i = \partial_v g_{i-1}(y_{i-1}, ui-1), \quad i \in \{1, \ldots, n\}$$

Vemos en esta forma de escritura que el sistema de estado es una relación de recurrencia no lineas con condición inicial y el sistema lineal adjunto es una relación de recurrencia retrograda con condición final.

Para calcular entonces $\bigtriangledown \tilde{J}(u)$ con el adjunto tendremos

$$\begin{pmatrix} \partial_v h_0(y_0, u_0)\lambda_1 \\ \partial_v h_1(y_1, u_1)\lambda_2 \\ \vdots \\ \partial_v h_{n-1}(y_{n-1}, u_{n-1})\lambda_{n-1} \end{pmatrix} + \begin{pmatrix} \partial_v g_0(y_0, u_0) \\ \partial_v g_1(y_1, u_1) \\ \vdots \\ \partial_v g_{n-1}(y_{n-1}, u_{n-1}) \end{pmatrix}$$

Usando el linealizado calcularemos $d_u y$ resolviendo

$$\begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ -\partial_z h_1 & 1 & 0 & \ldots & 0 \\ 0 & -\partial_z h_2 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & -\partial_z h_{n-1} & 1 \end{pmatrix} \begin{pmatrix} \partial_{u_0} y_1 & \ldots & \partial_{u_{n-1}} y_1 \\ \partial_{u_0} y_2 & \ldots & \partial_{u_{n-1}} y_2 \\ 0 & \ddots & \vdots \\ \partial_{u_0} y_n & \ldots & \partial_{u_{n-1}} y_n \end{pmatrix} =$$

$$\begin{pmatrix} -\partial_v h_0 & 0 & 0 & \ldots & 0 \\ 0 & -\partial_v h_1 & 0 & \ldots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & -\partial_v h_{n-2} & 0 \\ 0 & \ldots & 0 & 0 & -\partial_v h_{n-1} \end{pmatrix}$$

Se calcula el gradiente con la fórmula 3.11. Matricialmente será

$$d_u y^T \begin{pmatrix} \partial_z g_1 \\ \partial_z g_2 \\ \vdots \\ \partial_z g_{n-2} \\ \partial_z g_{n-1} \\ g_n' \end{pmatrix} + \begin{pmatrix} \partial_v g_0 \\ \partial_v g_1 \\ \vdots \\ \partial_v g_{n-2} \\ \partial_v g_{n-1} \end{pmatrix}$$

Veremos ahora la relación con un problema de control en sistemas con EDO. Dispondremos de nuevo de un dato inicial $y_0 \in \mathbb{R}$ y tanto la variable de estados como el control son funciones:

$$y : [0, T] \to \mathbb{R}$$

$$u : [0, T] \to \mathbb{R}$$

ligadas por el siguiente problema de valor inicial:

$$\begin{cases} \dot{y}(t) = \tilde{h}(t, y(t), u(t)), \\ y(0) = y_0. \end{cases} \tag{3.18}$$

Consideramos entonces el siguiente problema

$$\begin{cases} min \quad J(y, u) \\ s.a: \quad 3.18 \end{cases}$$

donde

$$J(y, u) = \tilde{\tilde{g}}(T, y(T)) + \int_0^T \tilde{g}((t, y(t), u(t))dt$$

Para resolverlo usando el caso que estudiamos más arriba consideramos un conjunto de tiempos discreto $0 = t_0 < t_1 < \ldots < t_n = T$ con $t_i = i \triangle t$, $\triangle t = \frac{T}{n}$, y aproximamos la solución de la EDO mediante el método de Euler Explicito:

$$\begin{cases} y_0 \in \mathbb{R} \text{ dado}, \\ y_{i+1} = y_i + \triangle t \tilde{h}(t_i, y_i, u_i). \end{cases}$$

Además debemos aproximar la integral del funcional coste mediante la regla del rectángulo retrógrada:

$$J_d(\overrightarrow{y}, \overrightarrow{u}) = \tilde{\tilde{g}}(T, y_n) + \sum_{i=0}^{n-1} \tilde{g}(t_i, y_i, u_i) \triangle t$$

Relacionándolo con el caso estudiado tenemos

$$\begin{cases} h_i(y_i, u_i) = y_i + \triangle t\tilde{h}(t_i, y_i, u_i), & i \in \{0, \dots, n-1\} \\ g_i(y_i, u_i) = \tilde{g}(t_i, y_i, u_i) \triangle t, & i \in \{0, \dots, n-1\} \\ g_n(y_n) = \tilde{\tilde{g}}(t_n, y_n) \end{cases}$$

El sistema adjunto sería

$$\begin{cases} \lambda_n = \tilde{\tilde{g}}'_n(T, y_n) \\ \lambda_i = \lambda_{i+1} + \triangle t\partial_z \tilde{f}(t_i, y_i, u_i)\lambda_{i+1} + \triangle t\partial_z \tilde{g}(t_i, y_i, u_i), & i \in \{n-1, \dots, 1\} \end{cases}$$

Lo cual se puede interpretar como una aproximación por Euler explicito, usado de manera retrograda de:

$$\begin{cases} \lambda(T) = \tilde{\tilde{g}}'_n(T, y_n) \\ \dfrac{d\lambda}{dt}(t) = \partial_y \tilde{f}(t, y(t), u(t))\lambda(t) + \partial_y \tilde{g}(t, y(t), u(t)) \end{cases}$$

que es una EDO lineal no homogénea.

Bibliografía

[1] *Álgebra lineal y geometría*, 3ª edición, Eugenio Hernández Rodríguez, María Jesús Vázquez Gallo, María Ángeles Zurro Moro, PEARSON EDUCATION, S.A., Madrid, 2012.

[2] *Álgebra lineal y geometría cartesiana*, 3ª edición, Juan de Burgos Román, McGranw-Hill, 2006.

[3] *Apuntes de Análisis Matemático I*, María D. Acosta, Camilo Aparicio, Antonio Moreno, Universidad de Granada.

[4] *Optimization theory and methods. Nonlinear programming*, volumen 1, Wenyu Sun, Ya-Xiang Yuan, Springer, 2006.

[5] *Numerical Optimization*, Nocedal, J., Wright, S. J., Springer-Verlag, 1999.

[6] *Optimización dinámica*, E. Cerdá Tena, Prentice Hall, 2001.

[7] *Ingeniería de control moderna*, K. Ogata, Pearson-Prentice-Hall, 2010.

[8] *Commande optimale. Notes du cours A08*, E. Trélat, 2007/2008.

[9] Deep Learning: An Introduction for Applied Mathematicians de Catherine F. Higham y Desmond J. Higham publicado en Siam Review Vol. 61, No. 4, pp. 860–891